*open*FT V10.0 for UNIX Systems and Windows Systems

*open*FT-Script Interface

Edition December 2006

Comments… Suggestions… Corrections…

The User Documentation Department would like to know your opinion of this manual. Your feedback helps us optimize our documentation to suit your individual needs.

Fax forms for sending us your comments are included in the back of the manual.

There you will also find the addresses of the relevant User Documentation Department.

Certified documentation according to DIN EN ISO 9001:2000

To ensure a consistently high quality standard and user-friendliness, this documentation was created to meet the regulations of a quality management system which complies with the requirements of the standard DIN EN ISO 9001:2000.

cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

Copyright and Trademarks

Copyright © Fujitsu Siemens Computers GmbH 2006.

All rights reserved.
Delivery subject to availability; right of technical modifications reserved.

All hardware and software names used are trademarks of their respective manufacturers.

This manual is printed on paper treated with chlorine-free bleach.

Contents

1 Preface 7

1.1 Target group 7
1.2 Concept of *open*FT manuals for UNIX and Windows systems 8
1.3 Notational conventions 9
1.4 README files 9
1.5 Requirements for *open*FT-Script 10

2 *open*FT-Script Commands 11

2.1 Overview of the *open*FT-Script commands 11
2.2 ftcans - Cancelling an *open*FT-Script request 12
2.3 ftdels - Deleting an *open*FT-Script request 14
2.4 ftscript - Starting an *open*FT-Script request 16
2.5 ftshwact - Displaying the activity associated with an *open*FT-Script request 18
Description of the output 20
2.6 Display *open*FT-Script requests 25

3 Structure of an *open*FT script 29

3.1 Components of an *open*FT script 29
3.1.1 Activities 30
3.1.2 Context 31
3.1.3 Referencing 31
3.2 Specifying file and directory names 34
3.2.1 File name attributes 34
3.2.2 Directory name attributes 35
3.3 Error handling 36
3.3.1 "Normal" Ftscript error codes 37
3.3.2 "Severe" Ftscript error codes 38
3.3.3 Restart 39
3.4 Running an *open*FT script 40

4 *open*FT-Script statements (alphabetical list) 43

4.1 Syntax of the *open*FT-Script statements 43
4.2 baseDir 44
4.3 comment 45
4.4 context 46
4.5 createDirectory 47
4.6 deleteDirectory 49
4.7 deleteFile 51
4.8 directory 53
4.9 empty 54
4.10 executeScript 56
4.11 fault 59
4.12 faulthandler 62
case 64
default 64
4.13 file 66
4.14 foreach 67
4.15 ftscript 73
4.16 list 74
4.17 listDirectory 75
4.18 parallel 78
4.19 partner 80
4.19.1 processingAdmission 81
4.19.2 transferAdmission 82
ftacAdmission 82
userAdmission 82
4.20 script 84
4.21 sequence 86
4.22 transferFile 88
AutoDataSpec 94
fromLocalFile 97
fromLocalTmpFile 99
fromRemoteFile 102

remoteFailureScript 104
remoteSuccessScript 106
toLocalFile 107
toLocalTmpFile 108
toRemoteFile 110

5 Error messages 111

Glossary 139

Index 143

1 Preface

This document describes Version 1 of *open*FT-Script.

*open*FT-Script is a language for the description of multiple logically interdependent *open*FT requests. *open*FT-Script makes it possible to combine these requests to form a single request (Ftscript).

*open*FT-Script reduces the customer workload involved in monitoring sequential *open*FT requests and permits restarts in the event of a downtime.

*open*FT-Script uses XML notation.

You can use *open*FT-Script under Windows™ as well as under Unix systems.

1.1 Target group

This manual is intended for XML programmers who want to create *open*FT-Script requests. *open*FT-Script requests are used to start *open*FT requests in Windows or Unix systems and, for example, transfer files to or from other systems.

A knowledge of Windows and UNIX-based operating systems as well as an understanding of XML would be useful when reading this manual.

The manual applies to Windows, Sun Solaris systems and to portings to other UNIX platforms. The operating system-dependent differences are described in detail in the Release Notice which is supplied with each product CD.

1.2 Concept of *open*FT manuals for UNIX and Windows systems

The complete description of *open*FT and its components comprises a number of different manuals. Alongside the present manual there are also other *open*FT manuals for Unix systems and Windows systems.
This description is distributed across the manuals as follows:

- *open*FT for UNIX systems - Installation and Administration

  The system administrator manual is intended for FT and FTAC administrators. It describes:
  – the installation of *open*FT and its optional components,
  – the operation, control and monitoring of the FT system and the FTAC environment,
  – the administration commands for FT and FTAC administrators
  – important CMX commands.

- *open*FT for UNIX systems - Enterprise File Transfer in the Open World

  The user manual is intended for the *open*FT user and describes:
  – the basic functions of the *open*FT product family,
  – the conventions for file transfers to computers running different operating systems,
  – details on implementing FTAM,
  – the *open*FT user commands,
  – the *open*FT-Script commands
  – the BSFT interface,

- *open*FT for Windows systems - Enterprise File Transfer in the Open World

  The user manual is intended for the *open*FT user and describes:
  – the basic functions of the *open*FT product family,
  – the conventions for file transfers to computers running different operating systems,
  – details on implementing FTAM

- *open*FT for UNIX systems and Windows systems - Program Interface

  This manual is intended for C programmers and describes the C program interface on UNIX and Windows systems.

- *open*FT for UNIX systems and Windows systems - *open*FT-Script Interface

  This manual is intended for XML programmers and describes:
  – the *open*FT-Script commands
  – the XML statements for the *open*FT-Script interface

i Many of the functions described in the manuals are also available in the *open*FT graphical interface. A detailed online help system that describes the operation of all the dialogs in the graphical interface is supplied together with the graphical interface. The online help system also contains a complete description of the *open*FT commands.

1.3 Notational conventions

The following notational conventions are used throughout this manual:

`typewriter font`
: typewriter font is used to identify entries and examples.

italics
: In running text, commands, statements, names, variables and values are indicated by italic letters, e.g. file names and host names.

i indicates notes

Additional conventions are used for the command descriptions and the program interface.

1.4 README files

Information on any functional changes and additions to the current product version can be found in product-specific README files.

1.5 Requirements for *open*FT-Script

*open*FT-Script is supplied with *open*FT and requires *open*FT V10 to be installed on the executing host. All the addressed partners must use an FTAM-/*open*FT-compatible product for file transfer.

If *open*FT is not used then the restrictions described in the *open*FT manual apply.

At least Java JRE 1.4.2-07 is required for the Java runtime environment. On Windows systems, the extended language version of Java JRE is also required. This does not have to be explicitly installed if Java JDK is installed.

2 *open*FT-Script Commands

The *open*FT-Script commands are used to start and administer *open*FT-Script requests. The requests themselves are stored in a text file in the form of XML statements. These XML statements are described beginning on page 43.

2.1 Overview of the *open*FT-Script commands

Starting and ending *open*FT-Script requests

ftscript Starts an *open*FT-Script request

ftcans Cancels an *open*FT-Script request

ftdels Deletes an *open*FT-Script request

Displaying *open*FT-Script requests and *open*FT-Script activities

ftshws Displays *open*FT-Script requests

ftshwact Displays the activities of an *open*FT-Script request

FT administrators can also use the *ftsetjava* command to administer the link to the Java executable, see "*open*FT System Administrator Manual" and the online help system.

As administrator, you can view, cancel and delete all the *open*FT-Script requests in the system and view the activities associated with all the *open*FT-Script requests. Users without administrator rights can only administer their own *open*FT-Script requests.

2.2 ftcans - Cancelling an *open*FT-Script request

tcans allows you to cancel openFT-Script requests that have not yet been concluded. You can cancel either a specific openFT-Script request or all the openFT-Script requests for a user. This also cancels any file transfer requests started by the specified openFT-Script requests which are currently running. This may take a little time. The status of the openFT-Script request is then set to "cancelled" to prevent any restart.

If the openFT-Script request that is to be cancelled is currently being processed then the following message is output at stderr:

`ftcans: Cancellation request for ftscript id` *`ftscript id`* `started`

If the request has been started but not yet processed then the following message is sent to stderr:

`ftcans: ftscript id` *`ftscript id`* `cancelled.`

Format

ftcans -h |
[-u=<User ID 1..32>]
<ftscriptid> | @a

Description

-h Outputs the command syntax on screen. Any specifications after *-h* are ignored.

-u=User ID
User ID under which the search for the *open*FT-Script request that is to be cancelled is performed.

Only administrators may input a user ID.

The default value is the calling party's user ID.

ftscriptid
Identification of the openFT-Script request. This is output if the *open*FT-Script request is started via an ftscript command.

@a means that all the user's *open*FT-Script requests are to be cancelled.

Return code

0	OK
4	Syntax error
51	Error while outputting an Ftscript user
54	Ftscript ID not found
250	Internal error

2.3 ftdels - Deleting an *open*FT-Script request

The specified, completed *open*FT-Script request is deleted from the user's directory or all completed *open*FT-Script requests are deleted from the user's directory.

No more information is subsequently available for deleted requests. A *ftshws* or *ftshwact* command with this ftscriptid is rejected since it no longer exists.

Before an *open*FT-Script request can be deleted, it must have been completed, i.e. ftshws must indicate the status T, F or C.

Since *ftcans* is not a synchronous command, it may be necessary to wait for the status C (cancelled) to arise before a subsequent *ftdels*.

If no *ftdels* is issued for an *open*FT-Script request then this is automatically deleted when the retention period expires.

Format

ftdels -h |
[-u=<User ID 1..32>]
<ftscriptid> | @a

Description

-h Outputs the command syntax on screen. Any specifications after *-h* are ignored.

-u=User ID
User ID under which the search for the *open*FT-Script request that is to be deleted is performed.

Only administrators may input a user ID.

The default value is the calling party's user ID.

ftscriptid
Identification of the *open*FT-Script request. This is output if the *open*FT-Script request is started via an *ftscript* command.

@a means that all the user's *open*FT-Script completed requests are to be deleted.

Return code

0	OK
4	Syntax error
51	Error while outputting an Ftscript user
54	Ftscript ID not found
56	*open*FT-Script has not completed
250	Internal error

2.4 ftscript - Starting an *open*FT-Script request

The *ftscript* command checks the specified script file and executes the statements it includes. The script file must contain a valid XML document which corresponds to the schema for the *open*FT-Script interface. It must also be possible to read the file using the caller's ID. The maximum number of users who may be owner of *open*FT-Script requests is 1024. This includes requests that are terminated but not yet deleted.

If errors occur during verification then the script file is not started and the errors are output at stderr.

If the script file starts correctly then the following message is output at stderr:

`ftscript: started successfully. Id:` *ftscript id*

Information about the *open*FT-Script request is stored in the internal *open*FT user memory during execution and through to expiry of the retention period. As a consequence, users can view the output *ftscript-id* in order to obtain information about the status and success of the operation.

ftscript is restartable, i.e. the processing of the *open*FT-Script request is ensured even after a system failure.

Format

ftscript -h |
[-t]
<Ftscriptfilename>

Description

-h Outputs the command syntax on screen. Any specifications after *-h* are ignored.

-t Diagnostic information (a trace) is created.

Ftscriptfilename
Name of the script file which contains the XML statements for the *open*FT-Script request that is to be run.

Return code

0	OK
4	Syntax error
50	Ftscript process could not be started
52	Maximum number of Ftscript users (1024) exceeded
55	Ftscript ID not found
250	Internal error

2.5 ftshwact - Displaying the activity associated with an *open*FT-Script request

Outputs information about the individual *open*FT-Script requests.

Format

ftshwact -h |
[-csv]
[-a=<ID of the activity> | -d=<Level depth 1...> | -c=<Chapter>]
[-st=[W]|[R]|[T]|[F]|[K]|[D]|[C]]
[-u=<user ID 1..32>]
<ftscriptid>

Description

-h Outputs the command syntax on screen. Any specifications after *-h* are ignored.

-csv The information is output in CSV format.

-a=ID of the activity
Only the specified activity is displayed.

You may also indicate a specific instruction in a request.

An activity's ID can be determined using a preceding ftshwact command (without the *-a* option). This means that you can view the status of the activity later.

-d=Level depth
Depth of the levels to be displayed.

All activities whose *activity ID* is not greater than the specified level number are displayed. The level number is the number of index numbers separated by dots.

Examples:
from a request with activity IDs 1, 1.2, 1.2(1).1, 1.2(1).2, 1.2(2).1, 1.2(2).2 and1.3 the option *-d*=2 selects the activities with the activity IDs 1, 1.2 and 1.3.

-c=Chapter
: Chapter corresponding to the activities to be displayed.

  Those activities are output that are a level below the activity with the activity ID specified as the chapter.

  In the above example, these are -c=1: 1.2 and 1.3; for -c=1.2: 1.2(1).1, 1.2(1).2, 1.2(2).1 and 1.2(2).2.

-st=[W] | [R] | [T] | [F] | [K] | [D] | [C]
: Display activities with the specified status. You can specify multiple statuses one after the other, e.g. *-st=WRT*.

  Activity 1 is always output since it displays the execution status of the entire script.

-u=user ID
: User ID under which the specified request is searched for.

  Only administrators may input a user ID.

  The default value is the calling party's user ID.

ftscriptid
: Identification of the *open*FT-Script request. This is output if the *open*FT-Script request is started via an *ftscript* command.

Return code

0	OK
4	Syntax error
51	Error while outputting an Ftscript user
53	Ftscript section not found
54	Ftscript ID not found
250	Internal error

Description of the output

Output is possible in tabular form and in CSV format.

It should be noted that for activities which have not yet been started, the output from the *ftshwact* command is usually incomplete since the references present in the request have not yet been resolved and it is not therefore possible to enter all the desired output values. In particular, file and directory names in reference specifications are not fixed until runtime since they may be dependent on the operating system.

Output in table format

The processing level of the activities is displayed in four columns:

Id Unique identification of the activity within the request. This can be converted into an Xpath which mirrors the position of the activity in the tree which is statically predefined by the XML script.

Dynamic information is simply added for the *foreach* nodes (sequence number in the *foreach* loop).

For more detailed information, see the description of the XML statements for the *open*FT-Script interface.

Sta Status of the statement. The following status identifiers are possible:

W (waiting)	The activity has not yet been started.
R (running)	The activity has been started but has not yet been terminated.
T (terminated)	The activity has been terminated without errors.
F (failure)	The activity has been terminated with an error.
K (killed)	The activity was cancelled by means of a faulthandler or an *ftcans* command.
D (dead)	The activity no longer starts due to a previous error.

In the case of the *ftscript* activity (first activity in an *open*FT-Script request), a distinction is made between the following statuses:

I (interrupted)	The request was interrupted, e.g. due to a system crash.
C (cancelled)	The request was cancelled with *ftcans*.

X (cancelling) The request is currently being cancelled due to an *ftcans* command.

F (failure) Is only displayed for the *ftscript* activity if the error was not handled by a *faulthandler*.

In the case of activities with the status F and *faulthandler* activities, the cause of the error is output in clear text in an additional line.

Activity

Activity name. The names are based on the *open*FT-Script language but may be truncated in some cases, e.g. *faulthdlr* instead of *faulthandler*.

foreach is designated in accordance with the value of the execute attribute as *foreach_P* (parallel) or *foreach_S* (sequential).

TransferFile is designated as *sendFile* or *rcvFile* (=receive File) depending on the direction of transfer.

ActivityObject

The content of this column depends on the activity in question, see the table below.

Activity	**ActivityObject**	**Meaning**
ftscript	<scriptPath>	Complete path name of the original file with the XML statements.
empty	-	
foreach_P	<contextObject>	context object which assumes the value of the current list element
foreach_S	as foreach_P	
parallel	-	
sequence	-	
sendFile	Specifies the remote file in the following form:	
	<partner>!<file name>	Partner with file name if both are known.
	*unknown!<file name>	if the partner is not yet known.
	*unknown!*unknown	if both are not yet known.

Activity	ActivityObject	Meaning
sendFile (*cont.*)	<partner>!*ref(<contextId>)	if *contextId* = *foreach contextObject* and the resolution is not yet known because it has not yet been passed through.
	<file name>	in the case of requests which have already been started, this is the name specified in the FT request. In the case of requests which have not yet been started, this name is derived from the operating system-specific name specified in the XML file (e.g. unixName) and extended by the directory specifications.
rcvFile	as sendFile.	
deleteFile	specifies the remote file as in sendFile (with partner), if the file is local without partner:	
	<file name>	like *sendFile*, is determined from the FT request in the case of requests that have already been started, and from the XML file in the case of requests that have not yet started. A local file name would be output as an absolute file name in the case of a started request and as a relative path name in the case of an as yet unstarted request.
	*unknown!<file name>	if it is not known if the file is local when a file object is referenced.
createDir	<partner>! <directory-name>	Partner with directory name if both are known.
	*unknown! <directory-name>	if the partner is not yet known.
	*unknown!*unknown	if both are not yet known.

Activity	**ActivityObject**	**Meaning**
createDir (*cont.*)	<partner>! *ref(<contextID>)	if *contexId = foreach contextObject* and the resolution is not yet known because it has not yet been passed through.
	<directory-name>	if the directory is local. In this case, as for *sendFile*, the name for already started requests is determined from the FT request and for requests which have not yet been started, from the specifications in the XML file. A local file name would be output as an absolute file name in the case of a started request and as a relative path name in the case of an as yet unstarted request.
deleteDir	as createDir.	
listDir	as createDir.	
execScript	32 characters.	Contains the first 32 characters of the command that is to be executed. For security reasons, the user should make sure that the first 32 characters do not contain any confidential parameters.
fault	<faultcode>	Error code specified by the user.
faulthdl	<triggering activity id>: <special faultcode>; <general faultcode>	

Output in CSV format

```
Id;State;Activity;ActivityObject;Partner;AddInfo;nrElements;
StartTime;Error
```

The output contains the following information:

Id	See table format on page 20.
State	See table format on page 20.
Activity	See table format on page 21.
ActivityObject	See table format, enclosed in double quotes, otherwise: - the path name is output without partner specifications - only the *faultcodes* are output for the *faulthandler* activity.
Partner	In the case of path-related activities, the partner or partner specification that would be present in front of the path name in table format, enclosed in double quotes. Otherwise empty.
AddInfo	For *sendFile* and *rcvFile*: TID, enclosed in double quotes if the activity has already started. Otherwise empty. For *faulthdl*, the triggering *activity-Id* enclosed in double quotes. Otherwise empty.
nrElements	In the case of a started *foreach*: number of loop passes. In the case of a started *parallel* or *sequence*: number of sub-activities.
StartTime	Start time in the format yyyy-mm-dd hh:mm:ss
Error	In the case of requests with the status F, case of error in clear text enclosed by double quotes. Otherwise empty.

2.6 Display *open*FT-Script requests

Outputs information about the status of a user's *open*FT-Script requests. You can also specify a *ftscriptid* in order to select a specific *open*FT-Script request.

Format

ftshws -h |
[-csv]
[-t]
[-v]
[-st=[W]|[R]|[T]|[F]|[I]|[C]|[X]]
[-u=<user ID 1..32> | @a]
[<ftscriptid>]

Description

-h Outputs the command syntax on screen. Any specifications after *-h* are ignored.

-csv The information is output in CSV format.

-t The openFT-Script requests are displayed sorted on generation time, beginning with the last request.

By default, the requests are displayed in alphabetical order.

-v Diagnostic information is also output (verbose).

If *-v* is specified then, in the case of *open*FT-Script requests which terminate with an error, the cause of the error is output in a second line after the tabular information.

In CSV format, the *-v* option is ignored.

-st=[W] | [R] | [T] | [F] | [I] | [C] | [X]
displays openFT-Script requests with the specified status, see *Sta* field in "Output in table format" on page 26.

You can specify multiple statuses one after the other, e.g. *-st=WRT*.

-u=user ID | @a
User ID for which openFT-Script requests are output or under which the specified request is searched for.
Only administrators may specify or *@a* (all user IDs).

The default value is the calling party's user ID.

ftscriptid
: Identification of the *open*FT-Script request. This is output if the *open*FT-Script request is started via an *ftscript* command.

  By default, all the user's *open*FT-Script requests are displayed.

Return code

0	OK
4	Syntax error
51	Error while outputting an Ftscript user
54	Ftscript ID not found
250	Internal error

Output in table format

The processing level of the *open*FT-Script requests is displayed in four columns:

User User ID under which the request was started.

Ftscriptid
: Unique identification of the request. The identification is returned by the *ftscript* command.

Sta Indicates the processing status, where:

W (waiting)	The request has not yet been started.
R (running)	The request has been started but has not yet been terminated.
T (terminated)	The request has been terminated without errors.
F (failure)	The request has been terminated with errors.
I (interrupted)	The request was interrupted, e.g. due to a system crash.
C (cancelled)	The request was cancelled with an *ftcans* command.
X (cancelling)	The request is currently being cancelled due to an *ftcans* command.

FtscriptFileName
: Path name of the script file.

If the status F and the *-v* option are specified then the cause of the error is output in clear text in another column.

Output in CSV format

```
User;Ftscriptid;State;CreationTime;FtscriptFileName;Error;
```

The output contains the following information:

User	User ID under which the request was started.
Ftscriptid	Unique identification of the request. The identification is returned by the *ftscript* command.
State	See table format (Sta).
CreationTime	Time at which the *open*FT-Script request was created, in the format yyyy-mm-dd hh:mm:ss.
FtscriptFileName	Path name of the script file.
Error	Cause of error in clear text in the case of *open*FT-Script requests with status F, otherwise empty.

User, *Ftscriptid*, *FtscriptFileName* and, if applicable, *Error* are output enclosed in double quotes.

3 Structure of an *open*FT script

3.1 Components of an *open*FT script

An *open*FT script consists of activities. Each activity has a context. The context may also describe error handling mechanisms (*faulthandler*).

- **Activities** may take the form of instructions issued to *open*FT (e.g. *transferFile*, *deleteFile*) or instructions which control the workflow (e.g. *parallel*, *foreach*). The instructions are described in chapter “openFT-Script statements (alphabetical list)” on page 43.
- Files, directories, scripts and partners can be stored as context objects in the **context**. By means of **referencing**, it is possible to re-use this type of context object in the activity and the underlying activities provided that these do not possess a context object with the same name. Re-use is not possible outside of the activity in which the context object is defined.
- If a fault occurs within an activity then **error handling** can be used to supply an appropriate response. If error handling terminates correctly (i.e. without errors) then the associated activity is considered to have been completed successfully. Similarly, if error handling is terminated with an error then the activity is considered to be defective.

The general structure of an *open*FT script (*ftscript*) is as follows:

```
<ftscript version="1">
  <context>
    ContextObjects*
    faulthandler?
  </context>
  Activities+
</ftscript>
```

Every *ftscript* has the statement <*ftscript version="1"*> as its root element. The root element contains the following sub-elements:

– an (optional) context with context objects and a maximum of one *faulthandler*
– one or more activities which are executed in the specified sequence

For further information on the syntax, see section “Syntax of the openFT-Script statements” on page 43.

3.1.1 Activities

There are various types of activity

Internal activities

Internal activities consist of instructions sent to the Ftscript interpreter to control operation.

These include *ftscript*, *sequence*, *parallel*, *foreach*, *empty* and *fault*
(For a description, see chapter “openFT-Script statements (alphabetical list)” on page 43).

External activities

The external activities are statements issued to *open*FT instructing it to run the required functions.

These include *executeScript*, *transferFile*, *deleteFile*, *createDirectory*, *deleteDirectory* and *listDirectory*
(For a description, see chapter “openFT-Script statements (alphabetical list)” on page 43).

Parent and child activities

You can nest activities (XML syntax). As a result, activities are subdivided into parent and child activities.

```
<ftscript version="1">
  <parallel>
    <transferFile …/>
    <listDirectory …/>
    <foreach …>
      <deleteFile …/>
    </foreach>
  </parallel>
</ftscript>
```

ftscript is the root element.

The root element has a child element (*parallel*).
parallel has *ftscript* as parent element (or higher-level element).

parallel also has three child elements (*transferFile*, *listDirectory* and *foreach*). The *foreach* activity also has a *deleteFile* activity as a child element.

3.1.2 Context

An activity's context describes the context objects and error handling mechanisms (*faulthandler*). Using the "context object" language tool, you can specify an element, for example a partner, once in the *open*FT script and then re-use it whenever necessary. To do this, you reference the context object at the point at which it is to be used. Using appropriate referencing, it is also possible to combine the properties of multiple context objects.

Each context object has an ID which must be unique within the context. This ID is used to address (reference) the context object.

If a referenced context object is not found in the current context then a (recursive) search is performed in the higher-level contexts. Context objects other context objects with the same ID in higher-level contexts.

A context is always present even if it has not been defined. There are certain activities (*foreach*, *listDirectory*) which automatically incorporate data in a context.

The context objects are described in more detail in section “context” on page 46.

3.1.3 Referencing

*open*FT-Script also allows you to combine context objects.
For example, a partner (see section “partner” on page 80) does not have to be specified in full at every location. You can transfer a partner to a context object and re-use it via referencing.

The following rules apply to de-referencing:

1. A *ref* attribute references a context object with the specified ID and the object type of the context object (e.g. file, partner).
2. The search is continued in the parent context if no suitable object is found in the current object. If no suitable context object is found there then the search is continued in its parent context.
   Multi-level referencing is permitted.
3. If no suitable context object is found then the script is terminated with the error *ft_reference* (see section “Error handling” on page 36).
4. *ref*=*"A"* is permitted in a context object with *ID*=*"A"*. In this case, *ref* always refers to the parent context.

5. Circular references are nor permitted:

   obj1(ID="A", ref="B"); *obj2(ID="B", ref="A")* is not possible in one and the same context. The error *ft_reference* is output.

6. All the attributes and elements of the referenced element which are not present in the referenced element are taken over. Before being taken over, the referenced element is de-referenced on the basis of these rules.

Examples

1. Valid referencing

```
<?xml version="1.0" encoding="UTF-8"?>
<ftscript version="1">
  <context>
    <partner id="remote" name="WindowsP_1">
      <transferAdmission>
        <ftacAdmission ftacAdmission="FTACADM"/>
      </transferAdmission>
    </partner>
    <file id="pack" name="pack1.bin">
      <partner ref="remote"/>
      <directory name="frg_eis_01"/>
    </file>
  </context>
  <transferFile>
    <fromRemoteFile ref="pack"/>
    <toLocalFile name="pack1.bin">
      <directory name="frg_eis_01"/>
    </toLocalFile>
  </transferFile>
  <transferFile>
    <context>
      <partner id="remote" name="UnixP_1">
        <transferAdmission>
          <ftacAdmission ftacAdmission="FTACADM"/>
        </transferAdmission>
      </partner>
    </context>
    <fromRemoteFile ref="pack">
      <partner ref="remote"/>
    </fromRemoteFile>
    <toLocalFile name="pack2.bin">
      <directory name="frg_eis_01"/>
    </toLocalFile>
  </transferFile>
</ftscript>
```

The file object with the *ID="pack"* references a partner object with *ID="remote"*.
The partner *WindowsP_1* is used in the first *transferFile* activity.

The same file object *pack* is referenced in the second *transferFile* activity. However, the partner has been overwritten. Consequently, *UnixP_1*, which is defined in the context of the second *transferFile*, is addressed. This definition hides the Windows partner with *Id="remote"*.

If the partner was not overwritten then the Windows partner would be used since this is found in the *pack* object definition in the *ftscript* context. The partner (on the Unix system) from the context of the second *transferFile* would be ignored since it is no longer accessible from the definition *Id="pack"*.

2. Invalid referencing

```
<?xml version="1.0" encoding="UTF-8"?>
<ftscript version="1">
  <transferFile>
    <fromRemoteFile ref="pack"/>
    <toLocalFile name="pack1.bin">
      <directory name="frg_eis_02"/>
    </toLocalFile>
  </transferFile>
</ftscript>
```

The *pack* reference is not defined. The script outputs an error *ft_noRef*. This is not processed (no *faulthandler* defined). The script is terminated before execution of the *transferFile* activity. No restart is possible. You can use *ftshws* to display the reason for termination.

3.2 Specifying file and directory names

Many *open*FT-Script statements use file or directory names. These are specified in the attributes *name*, *bs2000Name*, *unixName*, *windowsName* and *zosName*. If no special characteristics are defined then the following definition applies.

If a partner is specified then its operating system must also be explicitly specified. Otherwise *open*FT-Script itself determines the operating system of the local computer.

3.2.1 File name attributes

If a file name starts with the pipe character ("|") then this has the special meaning of a generating (*fromLocalFile*, *fromRemoteFile*) or receiving (*toLocalFile*, *toRemoteFile*) program call (see the *open*FT user manual).

Name	Value	Meaning
`name?`	string | `""`	File name with specification of the subpath (see *open*FT user manual). *name* is used if no operating system is known or no operating system-specific name has been specified. The default value is an empty string. If *name* is not specified and the profile *partner-ftac* is linked to a fixed file name then this applies. Only the character "/" may be used as a separator (e.g. */C:/x/y*).
`bs2000Name?`	string	File name for BS2000. The name is used if it has been specified and the associated partner is a BS2000 system. An empty string must be explicitly specified. (e.g. *C:\system\hugo*).
`windowsName?`	string	File name for Windows. The name is used if it has been specified and the associated partner or the associated local system is a Windows system. An empty string must be explicitly specified. You can use the "/" notation or the Windows-specific notation.

Name	Value	Meaning
unixName?	string	File name for Unix based system. The name is used if it has been specified and the associated partner or the associated local system is a Unix system. An empty string must be explicitly specified. You may only use the "/" notation.
zosName?	string	File name for ZOS. The name is used if it has been specified and the associated partner is a ZOS system. An empty string must be explicitly specified.

3.2.2 Directory name attributes

Name	Value	Meaning
name?	string | ""	Directory name. *name* is used if no operating system is known or no operating system-specific name has been specified. The default value is an empty string. If *name* is not specified and the profile *partner-ftac* is linked to a fixed directory name then this applies. Only the character "/" may be used as a separator.
bs2000Name?	string	BS2000-specific addressing (see *open*FT User Guide).
windowsName?	string	Windows-specific path. The name is used if it has been specified and the associated partner or the associated local system is a Windows system. An empty string must be explicitly specified. You can use the "/" notation or the Windows-specific notation.
unixName?	string	Unix system specific path. The name is used if it has been specified and the associated partner or the associated local system is a Unix system. An empty string must be explicitly specified. You may only use the "/" notation.
zosName?	string	z/OS-specific addressing (see *open*FT User Guide).

3.3 Error handling

The *open*FT script is checked when it is read.
If errors are identified then the *open*FT script is not run. When *ftscript* is started, you see a corresponding error message and a return code (see section “ftscript - Starting an openFT-Script request” on page 16).

If the check is completed successfully then the *open*FT script is executed in another process.
If an error occurs during execution then the cause of the error is logged.

Every error message has a unique internal code and is assigned to a specific error code. The errors are assigned to one of the following two categories on the basis of the error codes:

– "normal" Ftscript error codes

– "severe" Ftscript error codes

The table listing all these error messages can be found in chapter “Error messages” on page 111.

The error codes assigned by *open*FT-Script always start with "ft_".
You can also use the *fault* activity (see section “fault” on page 59) to assign any other error codes. However, these may not begin with "ft_". These are always considered to be "normal" error codes.

The cause of the error can be displayed with *ftshwact*.

If no *faulthandler* exists for an error then *ftscript* is terminated with an error (status F). The status and cause of termination can be displayed using *ftshws*.

3.3.1 "Normal" Ftscript error codes

"Normal" errors relate to the objects which are to be transferred or to the involved computer.

The fault code can be intercepted by the *faulthandler (default)* (see section “default” on page 64).

error codes	Description
ft_access	It is not possible to access a file/directory/computer.
ft_admin	Administration error
ft_auth	Authentication error (incorrect ID/password/authorizations).
ft_cantCreate	It is not possible to create a file/directory.
ft_cantDelete	It is not possible to delete a file/directory.
ft_configuration	Configuration error.
ft_connection	A connection error has occurred.
ft_corrupt	A file does not correspond to the expected format.
ft_exist	A file/directory already exists.
ft_local	The command script was terminated in the local system with an error.
ft_localFileStructure	Error in the local file.
ft_notEmpty	A directory for deletion is not empty.
ft_notExist	A file/directory/CSS/file owner does not exist.
ft_remote	The command script was terminated in the remote system with an error.
ft_remoteFileStructure	Error in the remote file.

3.3.2 "Severe" Ftscript error codes

The "severe" errors are primarily caused by internal problems (e.g. insufficient storage space on the hard disk for administrative information) or script errors (e.g. unresolved references).

If a „severe" error occurs than a restart is only possible under certain conditions (see section “Restart” on page 39).

"Severe" errors cannot be intercepted by the *faulthandler (default)*. In this case, it is necessary to write an explicit *faulthandler (case)* (see section “case” on page 64).

Error codes	Description
`ft_abort`	Termination by the user.
`ft_error`	A general error has occurred.
`ft_notSupported`	The functionality is not supported.
`ft_panic`	A serious error has occurred.
`ft_paramError`	A parameter error has occurred.
`ft_paramTooLong`	A parameter lies outside the value range.
`ft_recoveryFailed`	Error during restart (see section “Restart” on page 39). It is not possible to intercept the error.
`ft_reference`	A reference is invalid (not present or circular).
`ft_resource`	A resource error has occurred. (e.g. not enough storage space)
`ft_syntax`	A syntax error has occurred.

3.3.3 Restart

Ftscripts can generally be restarted if, for example, they have been aborted due to a system crash. Restrictions apply only to the following activities:

– *executeScript*, if *repeatable=no* was specified

– *createDirectory*, if *faultIfExists* was specified

– *deleteFile* or *deleteDirectory* if *faultIfNotExists* was specified

If the *open*FT-Script request is aborted during the processing of the statement then it is not possible to determine whether the activity has been completed. In the above cases it is not clear, when the restart is performed, how the *ftscript* should continue to run.

If, for example, a directory that was to be created already exists then it is not possible to determine whether it was created by the aborted *createDirectory* activity or whether it already existed before the Ftscript request was run.

If the restart operation encounters this type of ambiguous situation then it reacts as follows:

Activity	**Response on restart**
`executeScript` with the attribute `repeatable=no`	Activity aborted with the error *ft_resumeForbidden*
`createDirectory` with the attribute `faultIfExists`	If the directory already exists then activity aborted with the error *ft_recoveryCreateDirectory*
`deleteFile` or `deleteDirectory` with the attribute `faultIfNotExists`	If the file does not (or no longer?) exist(s) then the activity is aborted with the error *ft_recoveryFailed*

This response may occur if the *open*FT instance has been switched. If the *open*FT instance is deleted then all running *open*FT-Script requests are interrupted. They restart when the instance is switched. In the above cases, processing waits for approximately 2 seconds for the end of the activity after interruption of the request. In the case of lengthy *executeScript* activities this may not be enough, with the result that this Ftscript is aborted with an error when a restart attempt is made.

3.4 Running an *open*FT script

When a script is run, each activity passes through the following states:

– initialization
– execution
– end
– (error)

Initialization

The context is provided.

Execution

In the case of external activities, the *open*FT functionality is executed.
In the case of internal activities, the corresponding statement is executed.

If an error occurs during the execution of an activity than an error (or *fault*) is output together with an error code. The activity switches to the "error" state.

End

The end of the activity is reached if execution is terminated without an error. Data may be displayed in the higher-level context.

The activity is terminated. Processing continues with the next activity.
If there are no further activities then the *ftscript* is terminated.

Error

The "error" state may be caused by:

– an error occurring during the execution of the activity itself
– an error in a child activity which is not intercepted by a *faulthandler*

A suitable *faulthandler* is searched for in the current context (see section “faulthandler” on page 62). The activity is replaced by the content of the *faulthandler*. In this case, the context objects of the activity are displayed in the *faulthandler*. All the activity's child activities are aborted and their contexts are lost. If these child activities have started file transfer requests then these are also aborted.

If no suitable *faulthandler* is found then the error is passed to the parent activity. The parent activity switches to the "error" state.

If no suitable *faulthandler* is found in any of the higher-level activities then the entire script is terminated.

Diagnostic information

You can activate a trace in order to conduct a precise analysis of the *ftscript* run (including restart, see section “Restart” on page 39):

```
ftscript -t
```

The trace logs every action in the request.

4 *open*FT-Script statements (alphabetical list)

This section describes the individual *open*FT-Script statements in alphabetical order:

– The use of the statement.

– The existing restrictions.

– The format describes the syntax of the statement. For an explanation of the syntax, see the following section “Syntax of the openFT-Script statements”.

– The statement's attributes, their values and the meaning of these values. Optional attributes are indicated by a "?".

– The examples illustrate the use of the statement.

4.1 Syntax of the *open*FT-Script statements

The *open*FT-Script statements are described in the following sections. The syntax of the *open*FT-Script statements is specified in the "Format" section in the description of each statement. The following symbols are used:

Symbol	Meaning
`a?`	No element or an element a. Optional attributes are also indicated by a "?".
`a*`	No element or any number of elements a.
`a+`	One element a or multiple elements a.
`|` `a|b+`	Either ... or ... Either (exactly) one element a or one or more elements b.
`<...>`	The current element with its attributes is presented in angle brackets. Elements can be combined to form an activity. Such combined elements are printed without the angle brackets.

4.2 baseDir

You use *baseDir* to define a base directory for those *open*FT-Script statements that refer to a driectory, i.e.:

– *createDirectory* (see page 47),

– *deleteDirectory* (see page 49),

– *listDirectory* (see page 75).

When executing one of these statements the name specified or referrred with the statement is preceded with *baseDir* separated by a path separator.

baseDir should comprise the absolute or relative directory name after possible references have been resolved. Therefore the name specified with the statement should be the name of a subdirectory of this directory.

Restrictions

You can only use *baseDir* with the *open*FT-Script statements which are listed above.

Format

See the corresponding *open*FT-Script statement.

Attributes

Name	Value	Meaning
`dirnames`	See section “Directory name attributes” on page 35.	

Examples

See:
– section “createDirectory” on page 47.
– section “deleteDirectory” on page 49.
– section “listDirectory” on page 75.

4.3 comment

You use *comment* to enter a comment text for the element in question.

You can use comments to describe the scripts.

You use XML comments (<!--...-->) to make internal comments regarding the scripts.

Restrictions

There are no restrictions.

Format

```
<comment>
  text
</comment>
```

Attributes

There are no attributes.

4.4 context

You use *context* to define a context and, possibly, *faulthandlers* for an activity.

Context objects are objects which can be used in the same or in lower-level activities when referenced using the attribute *ref* or *listRef*. They may be of type *autoDataSpec*, *directory*, *file*, *list*, *partner* or *script*.

A context exists for every activity. If no context element is specified the an empty context is explicitly created.

All the context objects and *faulthandlers* of the higher-level contexts are visible provided that they are not hidden by context objects or *faulthandlers* with the same name.

Restrictions

There are no restrictions.

Format

```
<context>
 ContextObject*
 faulthandler?
</context>
```

Attributes

There are no attributes.

4.5 createDirectory

You use *createDirectory* to create a directory If you do not specify a partner then the directory is created under the local user ID.

You can use *baseDir* (see section “baseDir” on page 44) to define a base directory under which the specified directory is created.

You should note the response on a restart (see section “Restart” on page 39).

Restrictions

1. If you specify a directory path in the name attribute then all the directories down to the lowest level must already exist.
2. If you specify a base directory then this must already exist.
3. The length of the directory name (*baseDir* and *createDirectory*) is limited and depends on the *open*FT version. The length is the number of characters plus 1 character if *baseDir* does not end with a "/".

Format

```
<createDirectory ref?="ID" faultIfExists?="yes|no" dirnames >
  comment?
  context?
  partner?
  baseDir?
</createDirectory>
```

Attributes

Name	Value	Meaning
ref?	string	Reference to a directory context object
faultIfExists?	yes | no	The default value is *no*. If the directory exists then no message is output. If *yes* is specified then *ft_exists* is output if the directory already exists. On a restart (see section “Restart” on page 39), *ft_recoveryCreateDirectory* is output. This may also occur if the instance is switched.
dirnames	See section “Directory name attributes” on page 35.	

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<ftscript version="1">
  <context>
    <directory id="myDir" name="myTmp"/>
  </context>
  <createDirectory ref="myDir">
    <partner name="UnixP_1">
      <transferAdmission>
        <ftacAdmission ftacAdmission="FTACADM"/>
      </transferAdmission>
    </partner>
    <baseDir name="frg_eis_03"/>
  </createDirectory>
</ftscript>
```

Creates the directory *myTmp* for the FTAC transfer admission *FTACADM* in the directory *frg_eis_03* on the computer *UnixP_1*.

No error is output if the directory already exists.
Once *createDirectory* has run, the directory exists.

frg_eis_03 is specified as *baseDir*. The directory that is to be created is referenced. The entire directory path (*base* + *createDirectory*) is *frg_eis_03/myTmp* and consists of 16 characters.

If the desired directory *frg_eis_03/myTmp* cannot be created, for example because the path *frg_eis_03* does not exist then the error *ft_cantCreate* occurs.

A further example of *createDirectory* can be found in section “foreach” on page 67.

4.6 deleteDirectory

You use *deleteDirectory* to delete a directory. If you do not specify a partner then the directory is deleted under the local user ID.

You use *baseDir* (see section “baseDir” on page 44) to define a directory under which the directory that is to be deleted is searched for. The base directory name and the name specified with the directory name attribute are combined to form the directory name.

You should note the response on a restart (see section “Restart” on page 39).

Restrictions

1. The directory that is to be deleted must be empty.
2. The length of the directory name (*baseDir* and *delete Directory*) is limited and depends on the *open*FT version. The length is the number of characters plus 1 character if *baseDir* does not end with a "/".

Format

```
<deleteDirectory ref?="ID" faultIfNotExists?="yes|no" dirnames >
  comment?
  context?
  partner?
  baseDir?
</deleteDirectory>
```

Attributes

Name	Value	Meaning
ref?	string	Reference to a directory context object
faultIfNotExists?	yes | no	The default value is *no*. If the directory does not exist then no message is output. If *yes* is specified then *ft_notExists* is output if the directory does not exist. On a restart (see section “Restart” on page 39), *ft_recoveryFaile* is output. This may also occur if the instance is switched.
dirnames	See section “Directory name attributes” on page 35.	

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<ftscript version="1">
  <context>
    <directory id="myDir" name="myTmp"/>
  </context>
  <deleteDirectory ref="myDir">
    <partner name="UnixP_1" systemType="unix">
      <transferAdmission>
        <ftacAdmission ftacAdmission="FTACADM"/>
      </transferAdmission>
    </partner>
    <baseDir name="frg_eis_04"/>
  </deleteDirectory>
</ftscript>
```

Deletes the directory *myTmp* for the FTAC transfer admission *FTACADM* on the computer *UnixP_1*.

No error is output if the directory does not exist or has already been deleted.

frg_eis_04 is specified as *baseDir*. The directory that is to be deleted is referenced. The entire directory path (*base* + *deleteDirectory*) is *frg_eis_04/myTmp* and consists of 16 characters.

4.7 deleteFile

You use *deleteFile* to delete a file. If you do not specify a partner then the file is deleted under the local user ID.

If you want to delete all the files in a directory, you should use *listDirectory* (see section “listDirectory” on page 75) together with *foreach* (see section “foreach” on page 67).

You can use *baseDir* (see section “baseDir” on page 44) to define a base directory from which *deleteFile* is run.

You should note the response on a restart (see section “Restart” on page 39).

Restrictions

The length of the file name (*baseDir* and *deleteFile*) is limited and depends on the *open*FT version. THe length is the number of characters plus 1 character if *baseDir* does not end with a "/".

Format

```
<deleteFile ref?="ID" faultIfNotExists?="yes|no" filenames >
  comment?
  context?
  partner?
  directory?
</deleteFile>
```

Attributes

Name	Value	Meaning
ref?	string	Reference to a file context object
faultIfNotExists?	yes | no	The default value is *no*. If the file does not exist then no message is output. If *yes* is specified then *ft_notExists* is output if the file does not exist. On a restart (see section “Restart” on page 39), *ft_recoveryFailed* is output. This may also occur if the instance is switched.
filenames	See section “File name attributes” on page 34.	

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<ftscript version="1">
  <deleteFile name="hugo.trash">
    <partner name="UnixP_1" systemType="unix">
      <transferAdmission>
        <ftacAdmission ftacAdmission="FTACADM"/>
      </transferAdmission>
    </partner>
    <directory name="frg_eis_05"/>
  </deleteFile>
</ftscript>
```

Deletes the file *hugo.trash* for the FTAC transfer admission *FTACADM* in the directory *frg_eis_05* on the computer *UnixP_1*.

Errors are not handled in this example and result in the *open*FT script being deleted.

4.8 directory

You use *directory* to define a directory path.

You specify the partner where you use the corresponding *directory* object.

Restrictions

There are no restrictions.

Format

```
<directory id="ID" ref?="ref" dirnames >
  comment?
</directory>
```

Attributes

Name	Value	Meaning
id	string	A unique ID in the current context. The context object is referenced under this ID.
ref?	string	Name of the directory context object. Data which is not present here is taken over from the directory context object after de-referencing.
dirnames	See section “Directory name attributes” on page 35.	

4.9 empty

The activity *empty* does nothing but is required for formal purposes in order to intercept an error or other actions in a *faulthandler* (see section “faulthandler” on page 62).

Restrictions

There are no restrictions.

Format

```
<empty>
  comment?
</empty>
```

Attributes

There are no attributes.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<ftscript version="1">
  <transferFile>
    <context>
      <faulthandler>
        <default>
          <empty/>
        </default>
      </faulthandler>
    </context>
    <fromRemoteFile name="pack1.bin">
      <partner name="someHost">
        <transferAdmission>
          <ftacAdmission ftacAdmission="FTACADM"/>
        </transferAdmission>
      </partner>
      <directory name="frg_eis_06"/>
    </fromRemoteFile>
    <toLocalFile name="target.bin">
      <directory name="frg_eis_06"/>
    </toLocalFile>
  </transferFile>
</ftscript>
```

If errors occur in *transferFile* then execution of the Ftscript continues nevertheless, i.e. the Ftscript request is terminated with status T.

In the case of "severe" errors (see section “"Severe" Ftscript error codes” on page 38), the Ftscript request is terminated with the corresponding error message since the *default faulthandler* is ineffective.

4.10 executeScript

You use *executeScript* to run a script.

The script is executed in the target system. If you do not specify a partner then the script is executed under the local user ID.

The following command interpreters are used:

Operating system	Command interpreter
Windows	System call, i.e. an executable file with the specified name is searched for. E.g. to execute a shell command, enter *cmd /c*.
Unix system	/bin/sh -c
ZOS	TSO
BS2000	SDF

Restrictions

1. A script (e.g. bs2000Script, unixScript) may only occupy one line and has a limited length. The length depends on the *open*FT version.
2. You must specify a script which is not empty for the addressed operating system.

Format

```
<executeScript ref?="ID" repeatable?="true|false" >
  comment?
  context?
  script?
  bs2000Script?
  unixScript?
  windowsScript?
  zosScript?
  Partner?
</executeScript>
```

Attributes

Name	Value	Meaning
`ref?`	string	Reference to a script context object
`repeatable?`	`true` | `false`	The default value is *true*. The script may be repeated on a restart. If *false* is specified: The script may not be repeated on a restart. On a restart, the entire *ftscript* is aborted if it is not possible to determine whether this script has been fully processed. It is only possible to switch the *open*FT instance during script execution in the case of scripts with short runtimes (see section “Restart” on page 39).

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<ftscript version="1">
  <context>
    <list id="partnerList">
      <partner name="WindowsP_1" systemType="windows">
        <transferAdmission>
          <ftacAdmission ftacAdmission="FTACADM"/>
        </transferAdmission>
      </partner>
      <partner name="UnixP_1" systemType="unix">
        <transferAdmission>
          <ftacAdmission ftacAdmission="FTACADM"/>
        </transferAdmission>
      </partner>
    </list>
  </context>
  <foreach listRef="partnerList" selectType="partner"
contextObject=

"partner">
    <executeScript>
      <unixScript><![CDATA[echo hello unix
>frg_eis_07/demo.txt]]>

</unixScript>
      <windowsScript><![CDATA[cmd /c echo 'hello windows'
>frg_eis_07\

demo.txt]]></windowsScript>
      <partner ref="partner"/>
    </executeScript>
  </foreach>
</ftscript>
```

An *executeScript* is run on the computers in the list *partnerList*.

Corresponding operating-specific scripts are executed depending on the operating system in question. It is important to specify the operating system in the partner definition.

4.11 fault

You use *fault* to cancel the parent activity (and all its running child activities) and output an error message.

An error with the specified *faultcode* (error code) is output.
The error is intercepted with the appropriate *faulthandler* (*default* or *case*) (see section “faulthandler” on page 62). The activity associated with the *faulthandler* is executed.

All file transfer requests that are still running are also cancelled (*ft_cancel*). This results in the execution of the *remoteFailureScript* (see section “remoteFailure-Script” on page 104).

If it is not possible to assign a *faulthandler* to the error code then the entire script is aborted.

Restrictions

There are no restrictions.

Format

```
<fault code="faultcode">
  comment?
</fault>
```

Attributes

Name	Value	Meaning
code	Text	The error code which can be intercepted in a *faulthandler*.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<ftscript version="1">
  <parallel>
    <context>
      <partner id="remote" name="UnixP_1">
        <transferAdmission>
          <ftacAdmission ftacAdmission="FTACADM"/>
        </transferAdmission>
      </partner>
      <faulthandler>
        <case code="intercept">
          <deleteFile name="target1.bin">
            <partner ref="remote"/>
            <directory name="frg_eis_08NotAvailable"/>
          </deleteFile>
          <deleteFile name="target2.bin">
            <partner ref="remote"/>
            <directory name="frg_eis_08"/>
          </deleteFile>
        </case>
      </faulthandler>
    </context>
    <transferFile>
      <context>
        <faulthandler>
          <default>
            <fault code="intercept"/>
          </default>
        </faulthandler>
      </context>
      <fromLocalFile name=
"W:/openFT/ftscript/Test/data/small/bin.mp3"/>
      <toRemoteFile name="target1.bin">
        <partner ref="remote"/>
        <directory name="frg_eis_08NotAvailable"/>
      </toRemoteFile>
    </transferFile>
```

Example (cont.)

```
    <transferFile>
      <context>
        <faulthandler>
          <default>
            <fault code="intercept"/>
          </default>
        </faulthandler>
      </context>
      <fromLocalFile name=
"W:/openFT/ftscript/Test/data/large/bin.mp3"/>
      <toRemoteFile name="target2.bin">
        <partner ref="remote"/>
        <directory name="frg_eis_08"/>
      </toRemoteFile>
    </transferFile>
  </parallel>
</ftscript>
```

Two file transfers are performed in parallel.
If an error occurs during one of the transfers then this is indicated by the error code *intercept*. This is intercepted in the *faulthandler*.
The other file transfer is cancelled (*ft_cancel*).
The two target files are deleted in the *faulthandler*. Any errors which occur are ignored.

Using this script either both files or neither of the files reach their destination.

If a *remoteFailureScript* is defined for the file transfer which is cancelled by means of *ft_cancel* then this may continue to run even when the *faulthandler* is already active. This may lead to problems.

4.12 faulthandler

The *faulthandler* is analyzed if an error occurred in the activity in which it is present or if the activity is switched to the "error" state due to a child activity (see section “Running an openFT script” on page 40)

The *faulthandler* is used if a *case* activity with the corresponding error code is defined in it or, in the absence of any "severe" error, a *default* activity is defined (see section ““Severe" Ftscript error codes” on page 38).

This *case* or *default* activity then replaces the activity in which the *faulthandler* is located, takes over its context objects and is then executed.

If an error occurs during the execution of the *case* or *default* activity, then the original *faulthandler* is ignored and a suitable *faulthandler* in the activity's own context or that of its parent activities is used.

Restrictions

There are no restrictions.

Format

```
<faulthandler>
  comment?
  (case* default) | (case+ default?)
</faulthandler>
```

Attributes

See section “case” on page 64.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<ftscript version="1">
  <context>
    <partner id="RemotePartner" name="D018S011">
      <transferAdmission>
        <ftacAdmission ftacAdmission="p11"/>
      </transferAdmission>
    </partner>
  </context>
  <transferFile>
    <context>
      <file id="source" name="source.bin"/>
      <file id="target" name="target.bin">
        <partner ref="RemotePartner"/>
      </file>
      <faulthandler>
        <default>
          <transferFile>
            <context>
              <partner id="RemotePartner" name="D018S022">
                <transferAdmission>
                  <ftacAdmission ftacAdmission="p11"/>
                </transferAdmission>
              </partner>
            </context>
            <fromLocalFile ref="source"/>
            <toRemoteFile ref="target"/>
          </transferFile>
        </default>
      </faulthandler>
    </context>
    <fromLocalFile ref="source"/>
    <toRemoteFile ref="target"/>
  </transferFile>
</ftscript>
```

In this example, the local file *source.bin* is to be copied to *D018S011/taget.bin*. If an implicit error occurs in *transferFile* then the file is copied to *D018S022/target.bin*. This overlays the context object with the Id *Remote Partner*. If another error occurs during this activity then the script is cancelled.

case

The activity described in *case* is executed if the current error code is found in its list of error codes. The *case* activity replaces the activity in which the *faulthandler* is located.

The context objects of the replaced activity are copied to the context of the *case* activity if still present. The contexts of the child activities (including those in which the error occurred) are no longer accessible.

For information on execution, see section “sequence” on page 86.

Format

```
<case code="codelist" >
  comment?
  context?
  Activity+
</case>
```

Attributes

Name	Value	Meaning
code	codelist	A list of error codes for which this *case* activity is to be executed. The individual error codes are separated from one another by spaces.

default

If the current error code is not found in any of the *case* elements then the *default* activity is executed.

This applies to all error codes with the exception of the "severe" Ftscript error codes (see section ““Severe" Ftscript error codes” on page 38), for which the *default* branch of the *faulthandler* is ignored.
The *default* activity replaces the activity in which the *faulthandler* is located.

The context objects of the replaced activity are copied to the context of the *default* activity if still present. The contexts of the child activities (including those in which the error occurred) are no longer accessible.

For information on execution, see section “sequence” on page 86.

Format

```
<default>
  comment?
  context?
  Activity*
</default>
```

Attributes

There are no attributes.

4.13 file

You use *file* to define a file.

Properties of a file:

– The file has a system-specific name.

– The file is located on a concrete system (*partner*) in a concrete directory (*directory*)

Restrictions

There are no restrictions.

Format

```
<file id="ID" ref?="ref" filenames >
  comment?
  partner?
  directory?
</file>
```

Attributes

Name	Value	Meaning
id	string	A unique ID in the current context. The context object is referenced under this ID.
ref?	string	Name of another file context object.
filenames	See section “File name attributes” on page 34.	

4.14 foreach

You use *foreach* to execute a sequence for each element in a list (see section “list” on page 74). *foreach* executes the child elements of each element in the selected list as a sequence.

You can specify whether the sequences are executed one after the other (in the same order as the list elements) or in parallel.

Restrictions

There are no restrictions.

Format

```
<foreach listRef="ID" contextObject="ID" execute?=
"parallel|sequential"
         selectType="file|partner|directory" direction?=
"forward|reverse" >
  comment?
  context?
  Activity+
</foreach
```

Attributes

Name	Value	Meaning
`listRef`	string	Name of a valid context object of type *list*.
`contextObject`	string	Name of the *foreach* context object which takes on the value of the current list element. This must not be defined in the *foreach* context. It is defined implicitly. The type of context object corresponds to the type set in the *selectType* attribute.
`execute?`	`parallel` | `sequential`	The default value is *sequential*. The sequences are executed one after the other. If *parallel* is specified then the sequences are started in parallel.
`selectType`	`partner` | `file` | `directory`	Filters the elements of the specified type from the list. Only the filtered elements are iterated.
`direction?`	`forward` | `reverse`	The default value is *forward*. The list is worked through forwards. If *reverse* is specified then the list is worked through backwards.

Examples

1. Distributing files

```
<?xml version="1.0" encoding="UTF-8"?>
<ftscript version="1">
  <context>
    <list id="FileList">
      <file name="bin.mp3"/>
      <file name="text.txt"/>
    </list>
    <list id="HostList">
      <partner name="UnixP_1" systemType="unix">
        <transferAdmission>
          <ftacAdmission ftacAdmission="FTACADM"/>
        </transferAdmission>
      </partner>
      <partner name="WindowsP_1" systemType="windows">
        <transferAdmission>
          <ftacAdmission ftacAdmission="FTACADM"/>
        </transferAdmission>
      </partner>
    </list>
  </context>
  <foreach listRef="HostList" selectType="partner"
           contextObject="partner" execute="parallel">
    <foreach listRef="FileList" selectType="file"
             contextObject="file" execute="parallel">
      <transferFile>
        <fromLocalFile ref="file">
          <directory name="W:/openFT/ftscript/Test/data/large"/>
          <autoDataSpec binPattern="*.mp3" charPattern="*.txt"/>
        </fromLocalFile>
        <toRemoteFile ref="file">
          <partner ref="partner"/>
          <directory name="frg_eis_09"/>
        </toRemoteFile>
      </transferFile>
    </foreach>
  </foreach>
</ftscript>
```

The files *bin.mp3* and *text.txt* are copied to two computers.

In the example, the lists of files and computers are defined as context objects. The file list can also be defined, for example, by means of a *listDirectory* (see section “listDirectory” on page 75).

A double *foreach* sequence is used. The external sequence works through all the computers and the inner sequence works through all the files. The connection to the computers takes place in parallel and the files are also worked through in parallel at each computer.

autoDataSpec differentiates between text and binary files (see section “AutoDataSpec” on page 94).

When the script is run, the files are distributed to all the computers. Since no *faulthandler* was used in the example, the script is terminated with an error.

2. Deleting the file tree

```
<?xml version="1.0" encoding="UTF-8"?>
<ftscript version="1">
  <context>
    <partner id="remote" name="UnixP_1">
      <transferAdmission>
        <ftacAdmission ftacAdmission="FTACADM"/>
      </transferAdmission>
    </partner>
  </context>
  <listDirectory name="frg_eis_10/*//*" listObject="Flist">
    <partner ref="remote"/>
  </listDirectory>
  <foreach listRef="Flist" selectType="file"
           contextObject="delFile" execute="parallel">
    <deleteFile ref="delFile">
      <partner ref="remote"/>
    </deleteFile>
  </foreach>
  <foreach listRef="Flist" selectType="directory"
           contextObject="delDir" execute="sequential"
                                  direction="reverse">
    <deleteDirectory ref="delDir">
      <partner ref="remote"/>
    </deleteDirectory>
  </foreach>
</ftscript>
```

In this example, everything in the directory *frg_eis_10* on the computer *UnixP_1* is deleted under the FTAC transfer admission *FTACADM*.

listDirectory (see section “listDirectory” on page 75) is used to determine all the files and directories recursively using the search pattern *//*. The sequence in which the directories are listed corresponds to the sequence required for their generation (i.e. the opposite sequence is required in order to delete them).

In the first *foreach* sequence, all the files are deleted in parallel. Non-existent files are ignored. An error during file deletion results in cancellation of the script.

In the second *foreach* sequence, the empty directories are deleted backwards because the directories to be deleted with *deleteDirectory* must be empty (see section “deleteDirectory” on page 49). Non-existent directories are ignored. Other errors result in the cancellation of the script.

When the script has run, the directory *frg_eis_10* on the computer *UnixP_1* is empty.

3. Copying a file tree

```
<?xml version="1.0" encoding="UTF-8"?>
<ftscript version="1">
  <context>
    <partner id="remote" name="UnixP_1">
      <transferAdmission>
        <ftacAdmission ftacAdmission="FTACADM"/>
      </transferAdmission>
    </partner>
  </context>
  <listDirectory name="*//*" listObject="Flist">
    <partner ref="remote"/>
    <baseDir name="frg_eis_11"/>
  </listDirectory>
  <foreach listRef="Flist" selectType="directory"
           contextObject="creDir" execute="sequential">
    <createDirectory ref="creDir">
      <baseDir name="frg_eis_11"/>
    </createDirectory>
  </foreach>
  <foreach listRef="Flist" selectType="file"
           contextObject="file" execute="parallel">
    <transferFile>
      <fromRemoteFile ref="file">
        <partner ref="remote"/>
        <directory name="frg_eis_11"/>
      </fromRemoteFile>
      <toLocalFile ref="file">
        <directory name="frg_eis_11"/>
      </toLocalFile>
    </transferFile>
  </foreach>
</ftscript>
```

In the directory *frg_eis_11* on the computer *UnixP_1*, the file tree *//* is copied to the directory *frg_eis_11* under the local ID.

In the first *foreach* sequence, all the necessary directories are copied sequentially using *createDirectory*.
ListDirectory returns the directories *a*, *a/b* and *a/b/c* in sequence for the directory *frg_eis_11/a/b/c* (*frg_eis_11* is itself defined as the base directory with *baseDir*). A directory cannot be created unless the parent directory exists.

In the second *foreach* sequence, the files are copied in parallel since all the target directories are now present.

4.15 ftscript

ftscript is the root element of the script.

The element always corresponds to a *sequence* activity (see section “sequence” on page 86).

Restrictions

There are no restrictions.

Format

```
<ftscript version="1">
  comment?
  context?
  Activity+
</ftscript>
```

Attributes

Name	Value	Meaning
version	1	Fixed value describing the Ftscript version.

Example

See any example in the current manual.

4.16 list

A list contains multiple elements of type *partner*, *directory* or *file*.

You can also generate a list using *listDirectory* (see section “listDirectory” on page 75).

foreach permits the iterative processing of the elements in the list (see section “foreach” on page 67).

Restrictions

There are no restrictions.

Format

```
<list id="ID" >
 comment?
 ( partner | directory | file )*
</list>
```

Attributes

Name	Value	Meaning
id	string	A unique ID in the current context. The context object is referenced under this ID.

Example

See section “foreach” on page 67.

4.17 listDirectory

You use *listDirectory* to list the files and directories. The located file or directory names are combined in a list. The list is displayed in the parent context in a context object with the specified *listObject-Id*. The context object is available there after execution of the *listDirectory* activity.
If an error occurs during the execution of *listDirectory* then the object is not available.

A *listDirectory* as a direct child element of a *parallel* activity does not return a usable event list (see section “parallel” on page 78).

You can use *baseDir* (see section “baseDir” on page 44) to define a base directory under which the directory that is to be listed is searched for. The base directory name and the name specified with the directory name attribute are combined to form the directory name.
The base directory itself is not listed in the result.

Restrictions

1. You specify the name of the *list* context object which is displayed in the parent context. No context element with this name may exist.
2. The length of the directory name (*baseDir* and *listDirectory*) is limited and depends on the *open*FT version. The length is the number of characters plus 1 character if *baseDir* does not end with a "/".
3. You may only use the wildcard symbol **//** to list a directory tree in full.

Format

```
<listDirectory listObject="ID" dirnames >
  comment?
  context?
  partner?
  baseDir?
</listDirectory>
```

Attributes

Name	Value	Meaning
`ref?`	string	Reference to a directory context object
`listObject`	string	Name of the *list* context object which is displayed in the parent context. No context element with this name may exist.
`dirnames`	See section “Directory name attributes” on page 35. To list a directory tree in full, you can specify the wildcard symbol *//* at the end of the name.	

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<ftscript version="1">
  <context>
    <list id="RemoteHostList">
      <partner name="UnixP_1">
        <transferAdmission>
          <ftacAdmission ftacAdmission="FTACADM"/>
        </transferAdmission>
      </partner>
      <partner name="WindowsP_1">
        <transferAdmission>
          <ftacAdmission ftacAdmission="FTACADM"/>
        </transferAdmission>
      </partner>
    </list>
  </context>
  <listDirectory listObject="FileList">
    <baseDir name="W:/openFT/ftscript/Test/multi"/>
  </listDirectory>
  <foreach listRef="RemoteHostList" selectType="partner"
           contextObject="RemoteHost" execute="parallel">
    <foreach listRef="FileList" selectType="file"
             contextObject="File" execute="parallel">
      <transferFile>
        <fromLocalFile ref="File">
          <directory name="W:/openFT/ftscript/Test/multi"/>
        </fromLocalFile>
        <toRemoteFile ref="File">
          <partner ref="RemoteHost"/>
          <directory name="frg_eis_12/local/bin"/>
        </toRemoteFile>
      </transferFile>
    </foreach>
  </foreach>
</ftscript>
```

In the example, all the files from the local directory *W:/openFT/ftscript/Test/multi/* are written to the context object *FileList*.
The list only contains the file names, not the directory itself.

The files are distributed in parallel to *frg_eis_12/local/bin/* on all the computers in the *RemoteHostList*.

The context object *FileList* from the *listDirectory* activity is displayed in the context of the parent element (in the example, *ftscript*).

4.18 parallel

You can specify *parallel* to run all the activities "simultaneously" and independently of one another. However, truly simultaneous execution (e.g. time slicing) is not implemented.

The *parallel* activity is terminated when all the child activities have terminated. If one child activity outputs an error then the other child activities are cancelled.

The results of any given child activity are not visible in the other child activities. They are displayed in the context of the *parallel* activity when execution of all the child activities has completed.

The transfer of context objects to the parent context is not planned at present. Consequently, context objects which arise during the *parallel* activity are discarded.

A *listDirectory* as a direct child element of a *parallel* activity does not return a usable event list (see section “listDirectory” on page 75).

Restrictions

If the maximum number of outbound connections in *open*FT is set to "1" then all the file transfer requests are worked through sequentially. In this case, the *parallel* activity has no effect.

Performance:
If slow connections are used, a *parallel* activity may sometimes take longer than a *sequence* activity. The number of parallel connections is limited depending on the setting in *open*FT. The requests are buffered in the request queue. The work involved in administering the request queue may be greater than the corresponding sequential processing.

Format

```
<parallel>
  comment?
  context?
  Activity+
</parallel>
```

Attributes

There are no attributes.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<ftscript version="1">
  <parallel>
    <transferFile>
      <fromLocalFile name=
"W:/openFT/ftscript/Test/data/large/text.txt"

data="char"/>
      <toRemoteFile name="text.txt">
        <partner name="UnixP_1">
          <transferAdmission>
            <ftacAdmission ftacAdmission="FTACADM"/>
          </transferAdmission>
        </partner>
        <directory name="frg_eis_13"/>
      </toRemoteFile>
    </transferFile>
    <transferFile>
      <fromLocalFile name=
"W:/openFT/ftscript/Test/data/large/bin.mp3"

data="bin"/>
      <toRemoteFile name="bin.mp3">
        <partner name="WindowsP_1">
          <transferAdmission>
            <ftacAdmission ftacAdmission="FTACADM"/>
          </transferAdmission>
        </partner>
        <directory name="frg_eis_13"/>
      </toRemoteFile>
    </transferFile>
  </parallel>
</ftscript>
```

The files *text.txt* and *bin.mp3* are delivered "simultaneously".

In fact, the file transfer requests are sent to *open*FT "simultaneously". If enough capacity is free then these requests are executed simultaneously.

4.19 partner

You use *partner* to specify the partner for which the activity applies or at which the file or directory is located.

If a partner is specified then this is always considered to be the remote system even if your own computer and own user ID are being addressed. If you do not specify a partner the activity refers to the current user ID on the local computer.

After de-referencing, the partner must possess a transfer admission (see section “transferAdmission” on page 82).

Format

```
<partner id="ID" ref?="ref" name="name" sytemType?=
"any|unix|windows|zos|bs2000" >
  comment?
  transferAdmission
  processingAdmission?
</partner>
```

Attributes

Name	Value	Meaning
id	string	A unique ID in the current context. The context object is referenced under this ID.
ref	string	References a context object of type partner. After de-referencing, the partner must possess a transfer admission (see section “transferAdmission” on page 82).
name	string	Name of the partner system TNS and DNS names are permitted. IPv4 and IPv6 addresses start with %IP (see *open*FT user manual).
systemType?	any | unix | windows | zos | bs2000	The default value is *any*. Specifies the partner's system type. The system type is not determined automatically. If no system type is specified then the general data (e.g. name) is used instead of the system-specific data (e.g. unixname).

4.19.1 processingAdmission

You use *processingAdmission* to assign the processing admission for scripts.

processingAdmission is not currently supported. The attribute can be specified but has no effect.

Format

```
<processingAdmission userId?="user" userAccount?="account"
                                                    userPassword?=
"passwort" >
  comment?
</processingAdmission>
```

Attributes

Name	Value	Meaning
userId?	string	User ID Does not have to be specified if the *transferAdmission* is used (section “transferAdmission” on page 82).
userAccount?	string	Account information
userPassword?	string	Password for the user ID.

4.19.2 transferAdmission

You use *transferAdmission* to assign the admission for transfer files.

Format

```
<transferAdmission>
  comment
  ftacAdmission | userAdmission
</transferAdmission>
```

ftacAdmission

You use *ftacAdmission* to assign the admission in the form of an FTAC transfer admission.

Format

```
<ftacAdmission ftacAdmission="ftac" />
```

Attributes

Name	Value	Meaning
`ftacAdmission`	string	FTAC transfer admission (see *open*FT user guide).

userAdmission

You use *userAdmission* to specify the admission in the form of the login/LOGON access data (user ID, password and account).

Avoid using *userAdmission* in this version. *open*FT-Script requires the password to be specified in plain text.
If possible, use the more reliable *ftacAdmission* instead.

Format

```
<userAdmission userId="userId" userAccount?="account"
                                                    userPassword?=
"password" >
  comment?
</userAdmission>
```

Attributes

Name	Value	Meaning
userId?	string	User ID
userAccount?	string	Account information
userPassword?	string	Password for the user ID.

4.20 script

You use *script* to specify a text string which is to be executed as an operating system command. A context object of type *script* can be referenced in the *executeScript* activity or by *remoteSuccessScript* or *remoteFailureScript*.

The command must be written in the operating system-specific syntax of each operating system. This means that in a *script* object, you can specify a different text string for each operating system.

*open*FT-Script selects the text string that is to be executed on the basis of the operating system specification in the partner definition or on the basis of the local operating system. If the operating system is unknown or no *script* matching the operating system is specified then the text string designated with *script* is executed.

Restrictions

1. A script (e.g. bs2000Script, unixScript) may only occupy one line and has a limited length. The length depends on the *open*FT version.
2. In this version, the script is not configurable.
3. A return value other than zero is interpreted as an error and results in an *ft_scriptError*.

Format

```
<script id="ID" ref?="ID" repeatable?=true|false" >
  comment?
  script?
  bs2000Script?
  unixScript?
  windowsScript?
  zosScript?
  partner?
</script>
```

Attributes

Name	Value	Meaning
id	string	A unique ID in the current context. The context object is referenced under this ID.
ref	string	ID of another script object.
repeatable?	true | false	The default value is *true*. The script may be repeated on a restart. If *false* is specified then the script may not be repeated. If, on restart, it is not possible to determine whether the script has run then the entire *ftscript* is cancelled.

Examples

See section “executeScript” on page 56.

4.21 sequence

You use *sequence* to execute the activities in a sequence one after the other in the specified order.

Each child activity can use the results of the preceding child activities. If an activity adds new context objects to the *sequence* context (e.g. *transferFile/toLocalTmpFile* or *listDirectory*), then the following activities can access the new data.

Restrictions

There are no restrictions.

Format

```
<sequence>
  comment?
  context?
  Activity+
</sequence>
```

Attributes

There are no attributes.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<ftscript version="1">
  <sequence>
    <transferFile>
      <fromLocalFile name=
       "W:/openFT/ftscript/Test/data/small/text.txt" data="char"/>
      <toRemoteFile name="text.txt">
        <partner name="UnixP_1">
          <transferAdmission>
            <ftacAdmission ftacAdmission="FTACADM"/>
          </transferAdmission>
        </partner>
        <directory name="frg_eis_14"/>
      </toRemoteFile>
    </transferFile>
    <transferFile>
      <fromLocalFile name=
        "W:/openFT/ftscript/Test/data/large/bin.mp3" data="bin"/>
      <toRemoteFile name="bin.mp3">
        <partner name="WindowsP_1">
          <transferAdmission>
            <ftacAdmission ftacAdmission="FTACADM"/>
          </transferAdmission>
        </partner>
        <directory name="frg_eis_14"/>
      </toRemoteFile>
    </transferFile>
  </sequence>
</ftscript>
```

The files *text.txt* and *bin.mp3* are delivered one after the other.

First of all, the first request in the request queue is submitted. Once this has been completed and if no error message is triggered, the second request in the request queue is submitted.

4.22 transferFile

You use *transferFile* to perform a file transfer. *transferFile* starts the file transfer and waits inside *ftscript* for the end of the file transfer.
The file transfer itself can be restarted.

If you specify *remoteSuccessScript* (see section “remoteSuccessScript” on page 106) or *remoteFailureScript* (see section “remoteFailureScript” on page 104) then the corresponding script is subsequently run on the remote computer.
For local scripts, you should use *faulthandlers* (see section “faulthandler” on page 62) or *executeScript* (see section “executeScript” on page 56).

Types of file transfer

You use *transferFile* to transfer files as follows:

- File transfers from "remote" to "local" (*fromRemoteFile toLocalFile*)
- File transfers from "local" to "remote" (*fromLocalFile toRemoteFile*)
- File transfers from "remote" to "remote" can be accomplished by means of two sequential *transferFile* activities, e.g in a *sequence* activity:
  - File transfer from "remote" to "localTmp" (*fromRemoteFile toLocalTmpFile*)
  - File transfer from "localTmp" to "remote" (*fromLocalTmpFile toRemoteFile*)

Restrictions

1. After de-referencing, *fromRemoteFile* and *toRemoteFile* must possess a partner specification.
2. The elements *fromLocalFile* and *toLocalFile* must not possess any partner specification after de-referencing.
3. You cannot use *transferFile* to perform any file transfers from "local" to "local".

To perform file transfers from "local" to "local", you must use a corresponding script (*copy*) or specify one of the two files as "remote" and specify the local computer as the partner (specify *transferAdmission*).

Format

```
<transferFile compress?="none|byteRep|zip" writeMode?=
"replace|new|extend" transparentMode?="true|false"
dataEncryption?=" yes|no|onlyDataIntegrity">
  comment?
  context?
  ( (fromRemoteFile   toLocalFile)    |
    (fromLocalFile    toRemoteFile)   |
    (fromRemoteFile   toLocalTmpFile) |
    (fromLocalTmpFile toRemoteFile)  )
  remoteSuccessScript?
  remoteFailureScript?
</transferFile>
```

Attributes

Name	Value	Meaning
compress?	none | byteRep | zip	The default value is *none*. The file is not compressed. If *byteRep* is specified then identical sequences of characters are compressed. If *zip* is specified then zip compression is used.
writeMode?	replace | new | extend	The default value is *replace*. If the file exists then it is overwritten. If the file does not exist, it is created. If *extend* is specified then the data is appended to the existing file. If the file does not exist, it is created. If *new* is specified then a new file is created. If a file with this name already exists then the activity is cancelled with the error *ft_exist*.
transparentMode?	true | false	The default value is *false*. The file is transferred as standard. If *true* is specified then a transparent transfer is performed (e.g. in the case of transfers from a BS2000 system to another BS2000 system via a Windows/Unix system).
dataEncryption?	yes | no | onlyDataIntegrity	The default value is *no*. The user data is not encrypted. If *yes* is specified then the user data is encrypted (for settings see the *open*FT user manual). If *onlyDataIntegrity* is specified then only the data integrity is checked.

Examples

1. File transfer with *remoteSuccessScript* / *remoteFailureScript*

```
<?xml version="1.0" encoding="UTF-8"?>
<ftscript version="1">
  <context>
    <script id="anything ok">
      <unixScript><![CDATA[echo anything ok
           >frg_eis_15/status.txt]]>
      </unixScript>
      <windowsScript>
         <![CDATA[cmd /c "echo anything ok"
           >frg_eis_15\status.txt]]>
      </windowsScript>
    </script>
    <script id="anything crap">
      <unixScript><![CDATA[echo anything
           crap>frg_eis_15/status.txt]]>
      </unixScript>
      <windowsScript>
         <![CDATA[cmd /c echo anything crap
           >frg_eis_15\status.txt]]>
      </windowsScript>
    </script>
    <partner id="remote" name="UnixP_1" systemType="unix">
      <transferAdmission>
        <ftacAdmission ftacAdmission="FTACADM"/>
      </transferAdmission>
    </partner>
  </context>
  <sequence>
    <context>
      <faulthandler>
        <default>
          <executeScript ref="anything crap"/>
        </default>
      </faulthandler>
    </context>
    <transferFile>
      <fromRemoteFile name="bin.mp3">
        <partner ref="remote"/>
        <directory name="frg_eis_15"/>
      </fromRemoteFile>
      <toLocalFile name="bin.mp3">
        <directory name="frg_eis_15"/>
      </toLocalFile>
      <remoteSuccessScript ref="anything ok"/>
      <remoteFailureScript ref="anything crap"/>
    </transferFile>
```

Example 1 (cont.)

```
 <executeScript ref="anything ok"/>
  </sequence>
<transferFile writeMode="extend">
    <fromRemoteFile name="status.txt" data="char">
      <partner ref="remote"/>
      <directory name="frg_eis_15"/>
    </fromRemoteFile>
    <toLocalFile name="status.txt">
      <directory name="frg_eis_15"/>
    </toLocalFile>
  </transferFile>
</ftscript>
```

The file *bin.mp3* is transferred from the partner *remote* to the local file *bin.mp3*. The file *status.txt* is then created with the following contents:

– if transfer is OK

```
anything ok
anything ok
```

– if an error occurs

```
anything crap
anything crap
```

If errors occur during the compilation of the file *status.txt* then the script is aborted.

2. File transfer from "remote" to "remote"

```
<?xml version="1.0" encoding="UTF-8"?>
<ftscript version="1">
  <context>
    <partner id="remote1" name="UnixP_1">
      <transferAdmission>
        <ftacAdmission ftacAdmission="FTACADM"/>
      </transferAdmission>
    </partner>
    <partner id="remote2" name="WindowsP_1">
      <transferAdmission>
        <ftacAdmission ftacAdmission="FTACADM"/>
      </transferAdmission>
    </partner>
  </context>
  <transferFile>
    <fromRemoteFile name="data.txt">
      <partner ref="remote1"/>
      <directory name="frg_eis_16"/>
    <autoDataSpec charPattern="*.txt" binPattern="*.dat *.mp3"/>
    </fromRemoteFile>
    <toLocalTmpFile id="tmp"/>
  </transferFile>
  <transferFile>
    <fromLocalTmpFile use="tmp"/>
    <toRemoteFile name="data.txt">
      <partner ref="remote2"/>
      <directory name="frg_eis_16"/>
    </toRemoteFile>
  </transferFile>
</ftscript>
```

In the example, the file *data.txt* is first copied from the partner *remote1* to a temporary file. The temporary file is given an internal name. The suffix of the temporary file corresponds to the suffix of the associated *fromRemoteFile* (here **.txt*).
Conversion is performed using *autoDataSpec charPattern* for character symbols because the filename suffix corresponds to the pattern *.txt* (see section “AutoDataSpec” on page 94). If the local system is a Windows system then the line ends are converted accordingly.

When the temporary file is transferred to the remote system *remote2*, the *autoDataSpec* settings made when the temporary file was created are taken over. They are inherited as well as *MaxRecSize* and the data properties of the *fromRemoteFile* element. If the local system is a Windows system then the data is not converted on the second transfer. In the case of a local Unix system, the reverse would be true. When the transfer to the local system is performed, no data is converted. Instead, the data is converted on the subsequent transfer to the remote system.

AutoDataSpec

You use *AutoDataSpec* to define the transfer mode for the file in which the element was specified.

If the file's data type is unknown then it is determined on the basis of the file name using pattern recognition.
The only pattern currently permitted is **.xxx* since only the file name suffix is checked.
The file name specified for the actual transfer is used.

If a Unix-specific file name and a Windows-specific file name are specified then the appropriate file name is analyzed depending on the partner in question.

The file types are described in the *open*FT user manual.

Data type	Format	Example
`char`	Text format	*.xml
`bin`	Binary format	*.doc in a Windows system
`user`	User format	

AutoDataSpec describes which pattern is assigned to which data type.
If multiple patterns for different file formats match a file name then the file type is determined in the sequence *bin*, *char*, *user*.

Restrictions

1. Currently the only pattern permitted is **.xxx*. Only the file name suffix is checked.
2. If an explicit data type is specified for the file after de-referencing then *AutoDataSpec* is not evaluated.

Format

```
<autoDataSpec ref?="ID"
              charPattern?="patternList" binPattern?="
patternList"
              userPattern?=" patternList" default?="char|bin|user"
>
  comment?
</autoDataSpec>
```

Attributes

Name	Value	Meaning
ref?	string	Reference to an *AutoDataSpec* context object
binPattern?	patternlist[1]	Wildcard pattern. If the pattern matches the specified file name then the file is transferred in *bin* format.
charPattern?	patternlist[1]	Wildcard pattern. If the pattern matches the specified file name then the file is transferred in *char* format.
userPattern?	patternlist[1]	Wildcard pattern. If the pattern matches the specified file name then the file is transferred in *user* format.
default?	char | bin | user	The default value is *bin*. Specifies the assumed data type if no pattern is found.

[1]patternlist is the list of patterns separated by spaces.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<ftscript version="1">
  <context>
    <list id="FileList">
      <file name="bin.mp3"/>
      <file name="text.txt"/>
    </list>
    <list id="HostList">
      <partner name="UnixP_1">
        <transferAdmission>
          <ftacAdmission ftacAdmission="FTACADM"/>
        </transferAdmission>
      </partner>
      <partner name="WindowsP_1">
        <transferAdmission>
          <ftacAdmission ftacAdmission="FTACADM"/>
        </transferAdmission>
      </partner>
    </list>
  </context>
  <foreach listRef="HostList" selectType="partner"
                   contextObject="host">
    <foreach listRef="FileList" selectType="file"
                     contextObject="file">
      <transferFile>
        <fromLocalFile ref="file">
          <directory name="W:/openFT/ftscript/Test/data/small"/>
          <autoDataSpec charPattern="*.txt"
                        userPattern="*.tab *.dat" default="bin"/>
        </fromLocalFile>
        <toRemoteFile ref="file">
          <partner ref="host"/>
          <directory name="frg_eis_17"/>
        </toRemoteFile>
      </transferFile>
    </foreach>
  </foreach>
</ftscript>
```

In the example, the file list *FileList* of local files is transferred to all the computers in the *HostList*. The file transfer modes are activated in accordance with the *autoDataSpec* specification. **.txt* files are converted in accordance with the target system type. All unknown file types are transferred in binary form. **.tab* and **.dat* are defined for *user* transfers.

fromLocalFile

You use *fromLocalFile* to specify the local source file for file transfer.

The transfer type (data type) can be derived from the file name by means of *autodataSpec* (see section “AutoDataSpec” on page 94).

Restrictions

1. The length of the file name (*directory* and *fromLocalFile*) is limited and depends on the *open*FT version. The length is the number of characters plus 1 character if *directory* does not end with a "/".
2. *fromLocalFile* must not contain any partner specification after de-referencing.

Format

```
<fromLocalFile ref?="ID" data?="auto|char|bin|user"
               recordFormat?="std|undef|var|fix"
               maxRecSize?="int" ccsname?="string" filenames >
  comment?
  directory?
  autoDataSpec?
</fromLocalFile
```

Attributes

Name	Value	Meaning
ref?	string	Reference to another file object *partner* and *directory* are taken over from this if you have not specified the elements here (see section “Referencing” on page 31).
data?	auto | char | bin | user	The default value is *auto*. Specifies the data type. If *char*, *bin* or *user* are specified then the *AutoDataSpec* specification is ignored (see section “AutoDataSpec” on page 94).
record-Format?	std | undef | var | fix	The default value is *std*. Specifies the record format. The standard *open*FT assignment applies (*data=bin* -> *undef*, otherwise *var*) If *undef* is specified then the record format is undefined, e.g. in the case of binary formats. If *var* is specified then the record format is variable, e.g. in the case of text formats (1 record 1 line; lines can be of different lengths). If *fix* is specified then the record format is fixed, e.g. f80 (with *recordFormat=fix* and *maxRecSize=80*.

Name	Value	Meaning
maxRecSize?	1-32756	Specifies the record size. By default, the *open*FT value applies (see *open*FT user manual). If *data=char* then *maxRecSize* specifies the length of a line (to the cr/lf). Files with a line length greater than 32756 bytes must be transferred with *data=bin*.
ccsname?	string max. 8 characters	Specifies the Coded Character Set (see *open*FT user manual).
filenames	See section “File name attributes” on page 34.	

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<ftscript version="1">
  <transferFile>
    <fromLocalFile name="bin.mp3">
      <directory name="W:/openFT/ftscript/Test/data/small"/>
      <autoDataSpec charPattern="*.txt" userPattern="*.tab *.dat"
                    default="bin"/>
    </fromLocalFile>
    <toRemoteFile name="bin.mp3">
      <partner name="UnixP_1">
        <transferAdmission>
          <ftacAdmission ftacAdmission="FTACADM"/>
        </transferAdmission>
      </partner>
      <directory name="frg_eis_18"/>
    </toRemoteFile>
  </transferFile>
</ftscript>
```

In the example, the local file *bin.mp3* is transferred to the remote system *UnixP_1* under the name *bin.mp3*.
No transfer mode or *autoDataSpec* is specified. The transfer mode is *bin* (default value of *autoDataSpec*).

fromLocalTmpFile

You use *fromLocalTmpFile* to specify a temporary source file for file transfer from "remote" to "remote" (see “Types of file transfer” on page 88).

The temporary file is simply a buffer under the local ID. This temporary file is deleted automatically as soon as it can no longer be used.

The *data*, *MaxRecSize*, *recordFormat*, *ccsname* and *autoDataSpec* specifications in the *fromRemoteFile toLocalTmpFile* activity apply implicitly. When a *TmpFile* is created from *fromRemoteFile* then these specifications are inherited.

Restrictions

You may only use *fromLocalTmpFile* after a *fromRemoteFile toLocalTmpFile* activity since the file *TmpFile* already exists and must be accessible in the current context. For more information, see section “fromRemoteFile” on page 102 and section “toLocalTmpFile” on page 108.

Format

```
<fromLocalTmpFile use="tmpID" />
```

Attributes

Name	Value	Meaning
use	string	Reference to a file *TmpFile* which can be accessed in the current context. This file must previously have been created with a *transferFile* using *toLocalTmpFile*.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<ftscript version="1">
  <context>
    <partner id="FileServer" name="UnixP_1">
      <transferAdmission>
        <ftacAdmission ftacAdmission="FTACADM"/>
      </transferAdmission>
    </partner>
    <list id="RemoteHostList">
      <partner name="UnixP_1">
        <transferAdmission>
          <ftacAdmission ftacAdmission="FTACADM"/>
        </transferAdmission>
      </partner>
      <partner name="WindowsP_1">
        <transferAdmission>
          <ftacAdmission ftacAdmission="FTACADM"/>
        </transferAdmission>
      </partner>
    </list>
  </context>
  <listDirectory listObject="FileList">
    <partner ref="FileServer"/>
    <baseDir name="frg_eis_19/datastore"/>
  </listDirectory>
  <foreach listRef="FileList" selectType="file"
           contextObject="File" execute="parallel">
    <transferFile>
      <fromRemoteFile ref="File">
        <partner ref="FileServer"/>
        <directory name="frg_eis_19/datastore"/>
      </fromRemoteFile>
      <toLocalTmpFile id="tmpFile"/>
    </transferFile>
    <foreach listRef="RemoteHostList" selectType="partner"
             contextObject="RemoteHost" execute="parallel">
      <transferFile>
        <fromLocalTmpFile use="tmpFile"/>
        <toRemoteFile ref="File">
          <partner ref="RemoteHost"/>
          <directory name="frg_eis_19/targetDir"/>
        </toRemoteFile>
      </transferFile>
    </foreach>
  </foreach>
</ftscript>
```

In the example, the files in the directory *frg_eis_19/Datastore* are copied to the relevant *frg_eis_19/targetDir* target directory on the various computers.

listDirectory is used to identify the files on the *FileServer*.

The first *foreach* activity works through all the files identified on the *FileServer*. Each file is copied to a temporary file *tmpFile*.
In the second *foreach* activity, each temporary file is copied to the target directory *frg_eis_19/targetDir* on the *RemoteHost*.

Finally, the associated temporary file *tmpFile* is deleted.

fromRemoteFile

You use *fromRemoteFile* to specify the remote source file for file transfer.

The transfer type (data type) can be derived from the file name by means of *AutodataSpec* (see section “AutoDataSpec” on page 94).

Restrictions

1. The length of the file name (*directory* and *fromRemoteFile*) is limited and depends on the *open*FT version. The length is the number of characters plus 1 character if *directory* does not end with a "/".
2. *fromRemoteFile* must contain a partner specification after de-referencing.

Format

```
<fromRemoteFile ref?="ID" data?="auto|char|bin|user"
                recordFormat?="std|undef|var|fix"
                maxRecSize?="int" ccsname?="string" filenames >
  comment?
  partner?
  directory?
  autoDataSpec?
</fromRemoteFile
```

Attributes

Name	Value	Meaning
ref?	string	Reference to another file object *partner* and *directory* are taken over from this if you have not specified the elements here (see section “Referencing” on page 31).
data?	auto | char | bin | user	The default value is *auto*. Specifies the data type. If *char*, *bin* or *user* are specified then the *AutoDataSpec* specification is ignored (see section “AutoDataSpec” on page 94).
recordFormat?	std | undef | var | fix	The default value is *std*. Specifies the record format. The standard *open*FT assignment applies (*data=bin* -> *undef*, otherwise *var*) If *undef* is specified then the record format is undefined, e.g. in the case of binary formats. If *var* is specified then the record format is variable, e.g. in the case of text formats (1 record 1 line; lines can be of different lengths). If *fix* is specified then the record format is fixed, e.g. f80 (with *recordFormat=fix* and *maxRecSize=80*.
maxRecSize?	1-32756	Specifies the record size. By default, the *open*FT value applies (see *open*FT user manual). If *data=char* then *maxRecSize* specifies the length of a line (to the cr/lf). Files with a line length greater than 32756 bytes must be transferred with *data=bin*.
ccsname?	string max. 8 characters	Specifies the Coded Character Set (see *open*FT user manual).
filenames	See section “File name attributes” on page 34.	

Example

See example on page 91.

remoteFailureScript

You use *remoteFailureScript* to execute a script on the remote system if data transfer fails (see section “script” on page 84).
The functionality *-rf* of the *open*FT request is used (see *ft* command in the *open*FT User Guide).

You should note that *remoteFailureScript* and a corresponding *faulthandler* in the *ftscript* (see section “faulthandler” on page 62) may affect one another since their execution is practically simultaneous (local and remote).

See also section “executeScript” on page 56.

Additional notes

The script is also executed if a running data transfer is aborted with a *fault* (internally, *ft_cancel* is issued if a *fault* occurs, see section “fault” on page 59 and the *open*FT user manual).

If a *fault* occurs then *remoteFailureScript* is only executed for the running transfers.

Restrictions

1. The *remoteFailureScript* activity uses the partner from the associated *transferFile* activity (see section “transferFile” on page 88).
2. The specification of *repeatable* in referenced script objects is ignored.

Format

```
<remoteFailureScript ref?="ID">
  comment?
  script?
  bs2000Script?
  unixScript?
  windowsScript?
  zosScript?
</remoteFailureScript>
```

Attributes

Name	Value	Meaning
`ref?`	string	Reference to a script context object. Any partner described there is not evaluated. The partner in the associated *transferFile* activity is used.

Example

See example on page 91.

remoteSuccessScript

You use *remoteSuccessScript* to execute a script on the remote system if data transfer succeeds (see section “script” on page 84).
The functionality *-rs* of the *open*FT request is used (see *ft* command in the *open*FT User Guide).

See also section “executeScript” on page 56.

Additional notes

The script is also executed if a running data transfer is aborted with a *fault* (internally, *ft_cancel* is issued if a *fault* occurs, see section “fault” on page 59 and the *open*FT user manual).

Restrictions

1. The *remoteSuccessScript* activity uses the partner from the associated *transferFile* activity (see section “transferFile” on page 88).
2. The specification of *repeatable* in referenced script objects is ignored.

Format

```
<remoteSuccessScript ref?="ID">
  comment?
  script?
  bs2000Script?
  unixScript?
  windowsScript?
  zosScript?
</remoteSuccessScript>
```

Attributes

Name	Value	Meaning
ref?	string	Reference to a script context object. Any partner described there is not evaluated. The partner in the associated *transferFile* activity is used.

Example

See example on page 91.

toLocalFile

You use *toLocalFile* to specify the local target file for file transfer.

See also section “file” on page 66.

Restrictions

1. The length of the file name (*directory* and *toLocalFile*) is limited and depends on the *open*FT version. The length is the number of characters plus 1 character if *directory* does not end with a "/".
2. *toLocalFile* must not contain any partner specification after de-referencing.

Format

```
<toLocalFile ref?="ID" ccsname?="string" filenames >
  comment?
  partner?
  directory?
</toLocalFile>
```

Attributes

Name	Value	Meaning
ref?	string	Reference to another file object
ccsname?	string max. 8 characters	Specifies the Coded Character Set (see *open*FT user manual).
filenames	See section “File name attributes” on page 34.	

Example

See example on page 91.

toLocalTmpFile

You use *toLocalTmpFile* to specify a temporary target file for file transfer from "remote" to "remote" (see "Types of file transfer" on page 88).

The temporary file is simply a buffer under the local ID. The file is assigned to a file object with the specified ID. This file object is displayed in the parent context of the current *transferFile* element and must not already exist there.

The file is given an internal name. The suffix of the file is determined from the suffix of the associated *fromRemoteFile*. If this does not have a suffix then the generated temporary file also has no suffix. This file is stored in user memory. If "Diskquota" is activated then this may be exceeded when temporary files are created. A corresponding error message (*ft_err_LOCERR_MEM*) is output.

The file object inherits the *data*, *MaxRecSize*, *recordFormat*, *ccsname* and *autoDataSpec* specifications in the associated *fromRemoteFile* activity. These are then re-used when the *fromLocalTmpFile* activity is called (see section "fromLocalTmpFile" on page 99).

The temporary file is deleted automatically as soon as the associated file object is exited.

See also section "file" on page 66.

Restrictions

1. The length of the file name (*directory* and *toLocalTmpFile*) is limited and depends on the *open*FT version. The length is the number of characters plus 1 character if *directory* does not end with a "/".
2. You can only access the generated temporary file with *use="tmpID"* from a *fromLocalTmpFile*.
3. The *transferFile fromRemoteFile* action must be concluded before you use *transferFile toRemoteFile* to access the temporary file.

Format

```
<toLocalTmpFile id="tmpID" />
```

Attributes

Name	Value	Meaning
`id`	string	ID of the temporary file in *transferFile* elements associated with the parent context. You can only access this ID with *use="tmpID"* from a *fromLocalTmpFile*.

Example

See example in section “fromLocalTmpFile” on page 100.

toRemoteFile

You use *toRemoteFile* to specify the remote target file for file transfer.

See also section “file” on page 66.

Restrictions

1. The length of the file name (*directory* and *toRemoteFile*) is limited and depends on the *open*FT version. The length is the number of characters plus 1 character if *directory* does not end with a "/".
2. *toRemoteFile* must contain a partner specification after de-referencing.

Format

```
<toRemoteFile ref?="ID" ccsname?="string" filenames >
  comment?
  partner?
  directory?
</toRemoteFile>
```

Attributes

Name	Value	Meaning
ref?	string	Reference to another file object
ccsname?	string max. 8 characters	Specifies the Coded Character Set (see *open*FT user manual).
filenames	See section “File name attributes” on page 34.	

Example

See example on page 91.

5 Error messages

Internal code	Error code	Description
ft_activeDirNotExists	ft_resource	Script request directory does not exist. Restart not possible.
ft_alarmException	ft_error	Error during alarm handling. Is ignored.
ft_alarmFailed	ft_error	Slow poll: An unhandled error has occurred.
ft_callFtRuntime	ft_error	Runtime error on FT call.
ft_callSecurity	ft_access	Access error during file deletion.
ft_cancelCmdError	ft_error	Abort terminated with unknown error.
ft_cancelError	ft_error	Error during abort.
ft_cancelUnexpectedState	ft_error	Request terminated but status incorrect. Abort expected.
ft_cantCreateJobListener	ft_resource	Interruption during communication with *open*FT.
ft_cantInitializeJob	ft_error	Request cannot be initialized.
ft_cantWriteLogData	ft_resource	Log data could not be written.
ft_circleRef	ft_reference	Circular reference not permitted.
ft_closeOrderQueue	ft_resource	Order queue could not be closed correctly.
ft_code1038	ft_ignore	Request <request ID> is being completed and can no longer be deleted.
ft_code1050	ft_resource	Ftscript process could not be started.
ft_code1052	ft_resource	Maximum number of Ftscript users exceeded
ft_code1053	ft_notExist	Ftscript section not found.
ft_code1054	ft_notExist	Ftscript ID not found.
ft_code1055	ft_notExist	Ftscript file not found.

Internal code	Error code	Description
ft_code1056	ft_cantDelete	Ftscript request has not been completed.
ft_code108	ft_connection	Request <request ID>. Remote system not accessible.
ft_code20	ft_notExist	<local file> not found.
ft_code2014	ft_paramError	No modification of file attributes demanded.
ft_code2015	ft_admin	*open*FT is not authorized to process requests for this user.
ft_code2016	ft_notEmpty	The directory <local file> is not empty.
ft_code2017	ft_localFile Structure	The file attributes do not correspond to the request parameters.
ft_code2018	ft_error	Attributes could not be modified.
ft_code2019	ft_cantCreate	<local file> could not be created.
ft_code2021	ft_notExist	CCS name not known.
ft_code2022	ft_notExist	Higher-level directory not found.
ft_code2023	ft_exist	<local file> already exists.
ft_code2024	ft_notSupported	Transfer of file generation groups not supported.
ft_code2025	ft_access	Error accessing <local file>.
ft_code2026	ft_paramError	Resulting file name <local file> too long
ft_code2027	ft_paramError	No file or directory name specified.
ft_code2028	ft_auth	Invalid management password.
ft_code2029	ft_access	<local file> not available.
ft_code2030	ft_notExist	Home directory not found.
ft_code2031	ft_access	Not possible to rename.
ft_code2032	ft_resource	Not enough storage space for <local file>.
ft_code2033	ft_notExist	File owner unknown.
ft_code2034	ft_auth	Invalid file password.

Internal code	Error code	Description
ft_code2036	ft_cantDelete	The file's retention period has not yet expired.
ft_code2037	ft_auth	<local file> is write-protected.
ft_code2038	ft_localFile Structure	File structure not supported.
ft_code2039	ft_syntax	Syntax error in resulting file name <local file>.
ft_code2040	ft_notSupported	Transparent file transfer not supported.
ft_code2042	ft_notSupported	Not possible to extend the file during transparent transfer.
ft_code2043	ft_auth	Access to <local file> prohibited
ft_code2044	ft_paramTooLong	Follow-up processing too long.
ft_code2045	ft_auth	Authorization for follow-up processing invalid.
ft_code2046	ft_auth	Local transfer admission invalid.
ft_code2047	ft_auth	Request rejected by local FTAC.
ft_code2048	ft_notSupported	Function for protocol <partner protocol type> not supported.
ft_code2049	ft_notSupported	Remote follow-up processing not supported.
ft_code2070	ft_admin	Request <request ID> *open*FT is no longer authorized to process requests for this user.
ft_code2071	ft_notSupported	Request <request ID>. User data encryption not installed.
ft_code2072	ft_abort	Request <request ID> has been deleted.
ft_code2073	ft_corrupt	Request <request ID>. Error during encryption.
ft_code2074	ft_cantCreate	Request <request ID>. <local file> could not be created.
ft_code2075	ft_notExist	Request <request ID>. Higher-level directory no longer found.

Internal code	Error code	Description
ft_code2076	ft_access	Request <request ID>. Error on <local file> input/output
ft_code2077	ft_access	Request <request ID>. File now locked against concurrent access.
ft_code2078	ft_access	Request <request ID>. <local file> no longer available.
ft_code2079	ft_notExist	Request <request ID>. <local file> no longer found.
ft_code2080	ft_notExist	Request <request ID>. Home directory no longer found.
ft_code2081	ft_resource	Request <request ID>. <local file> can no longer be assigned any space.
ft_code2082	ft_notExist	Request <request ID>. File owner no longer known.
ft_code2083	ft_error	Request <request ID>. Error during preprocessing/postprocessing.
ft_code2084	ft_error	Request <request ID>. Exit code <2> during preprocessing/postprocessing.
ft_code2085	ft_auth	Request <request ID>. File password no longer valid.
ft_code2086	ft_auth	Request <request ID>. <local file> is now write-protected.
ft_code2087	ft_localFile Structure	Request <request ID>. File structure error.
ft_code2088	ft_error	Request <request ID>. NDMS error <2>.
ft_code2089	ft_recoveryFailed	Request <request ID>. Recovery failed.
ft_code2090	ft_error	Request <request ID>. Error in completing file transfer.
ft_code2092	ft_auth	Request <request ID>. Access to <local file> is no longer permitted.

Internal code	Error code	Description
ft_code2093	ft_error	Request <request ID>. FTAM error <2>.
ft_code2094	ft_cantDelete	Request <request ID>. The file's retention period has not yet expired.
ft_code2095	ft_notSupported	Request <request ID>. Not possible to extend the file during transparent transfer.
ft_code2096	ft_notSupported	Request <request ID>. File structure not supported.
ft_code2109	ft_connection	Request <request ID>. Connection request rejected by local transport system.
ft_code2110	ft_connection	Request <request ID>. Data integrity checking has detected an error.
ft_code2111	ft_connection	Encryption/data integrity check not possible. Encryption is disabled.
ft_code2112	ft_connection	Request <request ID>. Data integrity check is not supported by partner.
ft_code2113	ft_connection	Request <request ID>. User data encryption not possible for this request.
ft_code2114	ft_connection	Request <request ID>. Local system ID rejected by remote system ('<partner>').
ft_code2115	ft_connection	Request <request ID>. Request interrupted by remote system.
ft_code2116	ft_connection	Local application '<1>' not defined.
ft_code2117	ft_connection	Local application '<1>' not available.
ft_code2118	ft_connection	Request <request ID>. Authentication of local system failed.

Internal code	Error code	Description
ft_code2119	ft_connection	Request <request ID>. Local system unknown in remote system.
ft_code2120	ft_connection	Remote system '<partner>' unknown.
ft_code2121	ft_connection	Request <request ID>. Authentication of partner failed.
ft_code2122	ft_connection	Request <request ID>. Connection rejected or disconnected. Cause <2>
ft_code2123	ft_connection	Request <request ID>. Error <2> on OSS call.
ft_code2124	ft_connection	Request <request ID>. No free transport connection.
ft_code2125	ft_connection	Request <request ID>. Transport connection lost.
ft_code2126	ft_connection	Request <request ID>. Transport system error. Error code <2>.
ft_code2127	ft_connection	Request <request ID>. No data traffic within <2> seconds.
ft_code2140	ft_admin	Request <request ID>. Remote system: *open*FT is not authorized to process requests for this user.
ft_code2141	ft_notEmpty	Request <request ID>. Remote system: The directory '<remote file>' is not empty.
ft_code2142	ft_remoteFile Structure	Request <request ID>. Remote system: The file attributes do not correspond to the request parameters.
ft_code2143	ft_access	Request <request ID>. Remote system: Attributes could not be modified.

Internal code	Error code	Description
ft_code2144	ft_cantCreate	Request <request ID>. Remote system: File/directory '<remote file>' could not be created.
ft_code2145	ft_notExist	Request <request ID>. Remote system: CCS name unknown or not supported.
ft_code2146	ft_notExist	Request <request ID>. Remote system: Higher-level directory not found.
ft_code2147	ft_exist	Request <request ID>. Remote system: File/directory '<remote file>' already exists.
ft_code2148	ft_notSupported	Request <request ID>. Remote system: Transfer of file generation groups not supported.
ft_code2149	ft_access	Request <request ID>. Remote system: Error accessing '<remote file>'.
ft_code2150	ft_syntax	Request <request ID>. Remote system: Resulting file name too long.
ft_code2151	ft_access	Request <request ID>. Remote system: File locked against concurrent access.
ft_code2152	ft_paramError	Request <request ID>. Remote system: No file or directory name specified.
ft_code2153	ft_auth	Request <request ID>. Remote system: Invalid management password.
ft_code2154	ft_access	Request <request ID>. Remote system: File/directory '<remote file>' not available
ft_code2155	ft_notExist	Request <request ID>. Remote system: File/directory '<remote file>' not found.

Internal code	Error code	Description
ft_code2156	ft_notExist	Request <request ID>. Remote system: Home directory not found.
ft_code2157	ft_access	Request <request ID>. Remote system: Not possible to rename.
ft_code2158	ft_resource	Request <request ID>. Remote system: Not enough storage space for '<remote file>'.
ft_code2159	ft_notExist	Request <request ID>. Remote system: File owner unknown.
ft_code2160	ft_auth	Request <request ID>. Remote system: Invalid file password.
ft_code2161	ft_cantDelete	Request <request ID>. Remote system: The file's retention period has not yet expired.
ft_code2162	ft_auth	Request <request ID>. Remote system: File/directory '<remote file>' is write-protected.
ft_code2163	ft_remoteFile Structure	Request <request ID>. Remote system: File structure not supported.
ft_code2164	ft_syntax	Request <request ID>. Remote system: Syntax error in resulting file name.
ft_code2165	ft_notSupported	Request <request ID>. Remote system: Transparent file transfer not supported.
ft_code2166	ft_notSupported	Request <request ID>. Remote system: Not possible to extend the file during trans-parent transfer.
ft_code2167	ft_auth	Request <request ID>. Remote system: Access to '<remote file>' prohibited

Internal code	Error code	Description
ft_code2168	ft_paramTooLong	Request <request ID>. Remote system: Follow-up processing too long.
ft_code2169	ft_auth	Request <request ID>. Remote system: Transfer admission invalid.
ft_code2170	ft_notSupported	Request <request ID>. Remote system: Function not supported.
ft_code2195	ft_admin	Request <request ID>. Remote system: *open*FT is no longer authorized to process requests for this user.
ft_code2196	ft_abort	Request <request ID> has been deleted in the remote system.
ft_code2197	ft_cantCreate	Request <request ID>. Remote system: File/directory '<remote file>' could not be created.
ft_code2198	ft_notExist	Request <request ID>. Remote system: Higher-level directory no longer found.
ft_code2199	ft_access	Request <request ID>. Remote system: Error on '<remote file>' input/output
ft_code2200	ft_access	Request <request ID>. Remote system: File now locked against concurrent access.
ft_code2201	ft_access	Request <request ID>. Remote system: File/directory '<remote file>' no longer available.
ft_code2202	ft_notExist	Request <request ID>. Remote system: File/directory '<remote file>' no longer found.
ft_code2203	ft_notExist	Request <request ID>. Remote system: Home directory no longer found.

Internal code	Error code	Description
ft_code2204	ft_resource	Request <request ID>. Remote system: File/directory '<remote file>' can no longer be assigned any space.
ft_code2205	ft_notExist	Request <request ID>. Remote system: File owner no longer known.
ft_code2206	ft_error	Request <request ID>. Remote system: Error during preprocessing/postprocessing.
ft_code2207	ft_error	Request <request ID>. Remote system: Exit code <2> during preprocessing/postprocessing.
ft_code2208	ft_auth	Request <request ID>. Remote system: File password no longer valid.
ft_code2209	ft_auth	Request <request ID>. Remote system: File/directory '<remote file>' is now write-protected.
ft_code2210	ft_remoteFile Structure	Request <request ID>. Remote system: File structure error.
ft_code2211	ft_error	Request <request ID>. Remote system: NDMS error <2>.
ft_code2212	ft_recoveryFailed	Request <request ID>. Remote system: Recovery failed.
ft_code2213	ft_resource	Request <request ID>. Remote system: Resource bottleneck.
ft_code2214	ft_auth	Request <request ID>. Remote system: Access to '<remote file>' is no longer permitted.
ft_code2215	ft_error	Request <request ID>. FTAM error <2>.

Internal code	Error code	Description
ft_code2216	ft_remoteFile Structure	Request <request ID>. Remote system: File structure not supported.
ft_code2217	ft_cantDelete	Request <request ID>. Remote system: The file's retention period has not yet expired.
ft_code2218	ft_notSupported	Request <request ID>. Remote system: Not possible to extend the file during trans-parent transfer.
ft_code2226	ft_error	Inconsistent monitor file contents.
ft_code2227	ft_error	Monitor file not used by *open*FT.
ft_code2228	ft_error	Monitor file not present.
ft_code236	ft_admin	Set instance '<1>' no longer found.
ft_code35	ft_access	File locked against concurrent access.
ft_code41	ft_resource	Request queue full.
ft_code700	ft_syntax	The parameters '<1>' and '<2>' may not both be specified at once.
ft_code701	ft_syntax	Input error.
ft_code702	ft_syntax	Parameter value '<1>' too long.
ft_code703	ft_syntax	Mandatory parameter missing.
ft_code704	ft_syntax	Mandatory parameter '<1>' missing.
ft_code705	ft_syntax	Parameter '<1>' specified more than once.
ft_code706	ft_syntax	Parameter '<1>' can only be specified together with '<2>'.
ft_code707	ft_syntax	Invalid parameter '<1>'.
ft_code708	ft_syntax	Range of values for parameter '<1>' not respected.
ft_code709	ft_syntax	Too many positional parameters.

Internal code	Error code	Description
ft_code710	ft_syntax	Incorrect parameter value '<1>'.
ft_code750	ft_syntax	Command not known.
ft_code751	ft_syntax	Command name ambiguous with regard to '<1>'.
ft_code752	ft_syntax	Closing bracket missing for operand '<1>'.
ft_code753	ft_syntax	Incorrect separator '<1>' after operand '<2>'.
ft_code755	ft_syntax	List value of operand '<1>' not compatible with data type '<2>'.
ft_code756	ft_syntax	Introductory operand value required for '<1>'.
ft_code757	ft_syntax	Value of operand '<1>' not compatible with data type '<2>'.
ft_code758	ft_syntax	Keyword value of operand '<1>' ambiguous with regard to '<2>'.
ft_code759	ft_syntax	Too many closing brackets.
ft_code760	ft_syntax	Required operand '<1>' not present.
ft_code762	ft_syntax	Operand name '<1>' ambiguous with regard to '<2>'.
ft_code763	ft_syntax	Operand '<1>' not known.
ft_code764	ft_syntax	Operand '<1>' specified more than once.
ft_code765	ft_syntax	Too many list elements for operand '<1>'.
ft_code766	ft_syntax	Too many positional operands.
ft_code767	ft_syntax	Too many positional operands for '<1>'.
ft_code780	ft_syntax	Internal error: Insufficient operand memory.
ft_code781	ft_syntax	Internal error: Structure nesting too deep.
ft_code790	ft_syntax	Available commands: '<1>'.
ft_code791	ft_syntax	Available list values: '<1>'.

Internal code	Error code	Description
ft_code792	ft_syntax	Available operands: '<1>'.
ft_code793	ft_syntax	Available values: '<1>'.
ft_code800	ft_error	Request <request ID>. Internal error. Monitor file not accessible.
ft_code801	ft_error	Request <request ID>. Internal error.
ft_code802	ft_error	Request <request ID>. Warning: Inconsistent monitor file '<2>' contents.
ft_code803	ft_error	Request <request ID>. Follow-up processing could not be started.
ft_code804	ft_error	Request <request ID>. Inconsistent request data.
ft_code850	ft_error	Internal error. Monitor file not accessible.
ft_code851	ft_error	Internal error.
ft_code852	ft_error	Internal error. Set instance '<1>' incompatible.
ft_code853	ft_error	Reloading error. Error code <1>.
ft_code854	ft_error	No longer possible to write logging records. Process terminated.
ft_code855	ft_error	No further storage space for internal files.
ft_code856	ft_error	Error on OPS output.
ft_code857	ft_error	Error in key file '<1>'.
ft_code858	ft_error	Internal error. Not possible to set/reset file locks.
ft_code859	ft_error	Function not supported due to teleservice restrictions.
ft_code860	ft_error	Teleservice restriction for FTAC due to FT.
ft_code861	ft_error	Teleservice restriction for <1>.

Internal code	Error code	Description
ft_code862	ft_error	Protocol stack '<1>' not installed.
ft_code999	ft_error	*open*FT panic <1>. Abnormal termination.
ft_compressModeIllegal	ft_error	Defective compress mode. Defective schema?
ft_compressModeOutOf Range	ft_error	Incorrect specification in compress mode. ftAPI version?
ft_contextCantCreate	ft_error	Context file could not be created.
ft_contextCantCreate2	ft_error	Context file could not be generated.
ft_createDirectoryFailed	ft_resource	Directory could not be created or not a directory.
ft_createDirectorySecurity	ft_access	Directory could not be created.
ft_createNoFilename	ft_paramError	No name specified on generation of a directory.
ft_createParamError	ft_error	Parameter error on creation of a directory.
ft_deleteFailed	ft_error	Local file/local directory could not be deleted.
ft_deleteNoFilename	ft_paramError	No directory or file name was specified for deletion.
ft_deleteOrderQueue	ft_resource	Order queue could not be deleted.
ft_deleteParamError	ft_error	Parameter error on deletion.
ft_deleteParents	ft_error	..' Directory will not be deleted.
ft_deleteSelf	ft_error	.' Directory will not be deleted.
ft_delparError	ft_error	Error setting delpar.
ft_directionModeOutOf Range	ft_error	Incorrect specification for direction. ftAPI version?
ft_directoryElement Unknown	ft_error	Schema and Ftscript do not match. There is no such directory element.
ft_doubleContextObject	ft_error	Context object initialized more than once
ft_doubleLocalTmpFileRef	ft_reference	ID for local temporary file already assigned in this context.

Internal code	Error code	Description
ft_emptyEventQueue	ft_error	Internal error: Empty queue.
ft_emptyParallelQueue	ft_error	Internal error: Empty queue.
ft_emptyTransferAdmission	ft_paramError	The transfer admission does not contain any user or FTAC admission.
ft_encryptionIllegal	ft_error	Incorrect specification for encryption. Defective schema?
ft_encryptionOutOfRange	ft_error	Incorrect specification for encryption. ftAPI version?
ft_err_CONNERR_NOCONN	ft_connection	No free transport connection.
ft_err_CONNERR_NOTAVAIL	ft_connection	The remote system is not accessible.
ft_err_CONNERR_UNKNOWN	ft_configuration	The remote system is unknown.
ft_err_INT_CRFILE	ft_error	Error during file generation.
ft_err_INT_FORK_errno	ft_error	Error on fork system call. errno is the value of the errno variable. This value is set by the defective system call.
ft_err_INT_INIT	ft_error	It is not possible to initialize the server.
ft_err_INT_INTERNAL_FN	"ft_panic	Other internal error: Function not supported.
ft_err_INT_INTERNAL_VERS	"ft_panic	Other internal error: Data structure version not supported.
ft_err_INT_MEM	ft_resource	Error on storage request.
ft_err_INT_OPEN_errno	ft_error	Error on open system call. errno is the value of the errno variable. This value is set by the defective system call.
ft_err_INT_OPENDIR_errno	ft_error	Error on opendir system call. errno is the value of the errno variable. This value is set by the defective system call.
ft_err_INT_PIPE_errno	ft_error	Error on pipe system call. errno is the value of the errno variable. This value is set by the defective system call.

Internal code	Error code	Description
ft_err_INT_READ_errno	ft_error	Error on read system call. errno is the value of the errno variable. This value is set by the defective system call.
ft_err_INT_RMFILE_errno	ft_error	Error on rmfile system call. errno is the value of the errno variable. This value is set by the defective system call.
ft_err_INT_SIGNAL_signal	ft_error	The command was interrupted by signal. signal indicates the signal which caused the interruption.
ft_err_INT_STAT_errno	ft_error	Error on stat system call. errno is the value of the errno variable. This value is set by the defective system call.
ft_err_INT_SYSTEM_errno	ft_error	Error on system system call. errno is the value of the errno variable. This value is set by the defective system call.
ft_err_INT_WRITE_errno	ft_error	Error on write system call. errno is the value of the errno variable. This value is set by the defective system call. If it was not possible to write all the bytes, errno has the value -1.
ft_err_LOCERR_EXIST	ft_exist	The local file already exists.
ft_err_LOCERR_FTAC	ft_access	The request was rejected by the local FTAC.
ft_err_LOCERR_FTC_exit-status	ft_panic	exit-status indicates the message number of the ftc command (see message 94 and message 95 in the *open*FT message table).
ft_err_LOCERR_INCONS	ft_localFile Structure	The local file is inconsistent.
ft_err_LOCERR_MEM	ft_resource	The local file is assigned no space.

Internal code	Error code	Description
ft_err_LOCERR_NOACCESS	ft_access	It is not possible to access the local file.
ft_err_LOCERR_NOCREAT	ft_cantCreate	The local file cannot be created.
ft_err_LOCERR_NOTEXIST	ft_notExist	The local file cannot be found.
ft_err_PAR_DIRAC_errno	ft_error	errno is the value of the errno variable. This value is set by the stat() call. The errno variable has the value 0 if no write authorization has been assigned for the directory.
ft_err_PAR_FTMSG_code	Error code in accordance with ft_code<code>	<Message in accordance with ft_code<code>>.
ft_err_PAR_INVSESS	ft_panic	The session number is invalid.
ft_err_PAR_LEN	ft_notSupported	The name of the working directory (workdir) is too long.
ft_err_PAR_LEN_ACCOUNT	ft_paramTooLong	Parameter too long. ftamext->account.
ft_err_PAR_LEN_CRPWD	ft_paramTooLong	Parameter too long. ftamext->crpasswd.
ft_err_PAR_LEN_FPWD	ft_paramTooLong	Parameter too long. mgmtpasswd or filepasswd.
ft_err_PAR_LEN_LEGALQ	ft_paramTooLong	Parameter too long. ftamext -> legalq.
ft_err_PAR_LEN_LOCFN	ft_paramTooLong	Parameter too long. locfn.
ft_err_PAR_LEN_LOCPR	ft_paramTooLong	Parameter too long. Total of the lengths of locsuccproc and locfailproc.
ft_err_PAR_LEN_REMACC	ft_paramTooLong	Parameter too long. remaccount
ft_err_PAR_LEN_REMADM	ft_paramTooLong	Parameter too long. remadmis.
ft_err_PAR_LEN_REMFN	ft_paramTooLong	Parameter too long. fn or remfn.
ft_err_PAR_LEN_REMPR	ft_paramTooLong	Parameter too long. Total of the lengths of remsuccproc and remfailproc.
ft_err_PAR_LEN_REMPWD	ft_paramTooLong	Parameter too long. rempasswd.
ft_err_PAR_LEN_REMSYS	ft_paramTooLong	Parameter too long. remsys.

Internal code	Error code	Description
ft_err_PAR_MAND	ft_panic	– The parameter list par was not specified. – The name of the working directory (workdir) was not specified. – The output area stat was not specified. The parameter list par was not specified. – The output area info was not specified (only in the case of ft_show()).
ft_err_PAR_MAND_LOCFN	ft_paramError	locfn was not specified.
ft_err_PAR_MAND_REMSYS	ft_error	The remote system was not specified.
ft_err_PAR_NODIR	ft_panic	The specified name (workdir) does not designate a directory.
ft_err_PAR_NOTERM	ft_panic	The request is not yet active.
ft_err_PAR_OPEN	ft_panic	In a program, the same directory (workdir) has already been assigned to a session.
ft_err_PAR_REMOTE_ NOACCESS	ft_access	No authorization to delete in the remote system. No authorization to read attributes in the remote system.
ft_err_PAR_REMOTE_ NOTEMPTY	ft_notEmpty	The directory in the remote system is not empty.
ft_err_PAR_REMOTE_ NOTEXIST	ft_notExist	File/directory does not exist in the remote system.
ft_err_PAR_TERM	ft_ignore	The request has already terminated.
ft_err_PAR_VALUE	ft_panic	Unknown parameters/parameters are incompatible. The name of the working directory (workdir) is invalid.
ft_err_PAR_VALUE_ACCESS	ft_panic	Invalid parameter: ftamext ->accessmode.
ft_err_PAR_VALUE_AVAIL	ft_panic	Invalid parameter: ftamext ->available.

Internal code	Error code	Description
ft_err_PAR_VALUE_CANTIME	ft_panic	Invalid parameter: cantime.
ft_err_PAR_VALUE_COMPR	ft_panic	Invalid parameter: compress.
ft_err_PAR_VALUE_DIR	ft_panic	Invalid parameter: direction.
ft_err_PAR_VALUE_ENCRYPT	ft_resource	Invalid parameter: encryption.
ft_err_PAR_VALUE_FPWD	ft_panic	Invalid parameter: mgmtpasswd.
ft_err_PAR_VALUE_FTYPE	ft_panic	Invalid parameter: filetype.
ft_err_PAR_VALUE_LOCCCSN	ft_paramError	Invalid parameter: locccsn.
ft_err_PAR_VALUE_MAXREC	ft_paramError	Invalid parameter: maxrecsize.
ft_err_PAR_VALUE_PRIO	ft_panic	Invalid parameter: priority.
ft_err_PAR_VALUE_REMACC	ft_paramError	Invalid parameter: remaccount
ft_err_PAR_VALUE_REMADM	ft_auth	Invalid parameter: remadmis or invalid parameter: remadm. The user ID/transfer admission in the remote system is invalid.
ft_err_PAR_VALUE_REMCCSN	ft_paramError	Invalid parameter: remccsn.
ft_err_PAR_VALUE_REMFN	ft_paramError	Invalid parameter: remfn. The specified file does not exist/access is not permitted.
ft_err_PAR_VALUE_REMPWD	ft_paramError	Invalid parameter: rempasswd.
ft_err_PAR_VALUE_REMSYS	ft_paramError	Invalid parameter: rem. The specified remote system is unknown.
ft_err_PAR_VALUE_RFORM	ft_paramError	Invalid parameter: record format.
ft_err_PAR_VALUE_RID	ft_panic	The request ID (rid) is invalid.
ft_err_PAR_VALUE_STARTTIME	ft_panic	Invalid parameter: starttime.
ft_err_PAR_VALUE_SYNC	ft_panic	Invalid parameter: synchron.
ft_err_PAR_VALUE_TRANSP	ft_panic	Invalid parameter: transparent.
ft_err_PAR_VALUE_WMODE	ft_panic	Invalid parameter: writemode.
ft_err_PAR_VERS	ft_panic	The version of the data structure (parameter list or output area) is invalid.
ft_err_REMERR_EXIST	ft_exist	The remote file already exists.
ft_err_REMERR_INCONS	ft_remoteFile Structure	The remote file is inconsistent.

Internal code	Error code	Description
ft_err_REMERR_MEM	ft_cantCreate	The remote file is assigned no space.
ft_err_REMERR_NOACCESS	ft_access	It is not possible to access the remote file.
ft_err_REMERR_NOCREAT	ft_cantCreate	The remote file cannot be created.
ft_err_REMERR_NOTEXIST	ft_notExist	The remote file cannot be found.
ft_err_REMERR_REMADM	ft_auth	The remote transfer admission is invalid.
ft_errorEncoding	ft_resource	Code Cp850 not supported. Install extended language support for Java.
ft_errorOnCancel	ft_error	Error during abort.
ft_errorReadingCsv	ft_resource	User information could not be read.
ft_errorReadingExec	ft_resource	Default error output from script execution could not be read.
ft_errorReadingFtsadmErr	ft_resource	Default error output from fsadm command could not be read.
ft_errorWaitingForExec	ft_resource	Error waiting for the end of script execution.
ft_errorWaitingForFtsadm	ft_resource	Error waiting for the end of fsadm command.
ft_execFailed	ft_error	Script execution could not be started.
ft_exist	ft_exist	Directory already exists.
ft_exit_<exitcode>	ft_remote	The command script was terminated in the remote system with the error <exitcode>.
ft_exit_255	ft_local	The command script was terminated in the local system with an error.
ft_featuresNotSupported	ft_error	The parser does not support necessary features.
ft_ftsadmFailed	ft_error	IO exception on start of ftsadm.
ft_housekeepingError	ft_error	Housekeeper terminated with unknown error. Is repeated.

Internal code	Error code	Description
ft_illegalJobState	ft_error	Invalid status found on end of request.
ft_illegalMode	ft_error	Invalid mode.
ft_infoFileCantWrite	ft_resource	Info file cannot be written.
ft_InterpreterNotAlive	ft_resource	Interpreter could not be started. Execution later.
ft_invalidStartState	ft_error	Invalid status on start.
ft_jobCantCreate	ft_resource	The request could not be generated.
ft_jobCloseError	ft_resource	Error terminating request.
ft_jobExists	ft_error	This request already exists.
ft_jobFailed	ft_error	An unknown error has occurred.
ft_listDirectoryError	ft_resource	Directory cannot be read.
ft_listDirectoryFileInfo	ft_resource	Error writing persistent data.
ft_listDirectoryNoDirectory	ft_notExist	No directory:
ft_listDirectoryOutOfRangeError	ft_error	Area error on ListDirectory.
ft_listDirectoryParamError	ft_error	Parameter error on ListDirectory.
ft_listDirectorySecurity	ft_access	Access error on ListDirectory for a local directory.
ft_listDirWriterCantCreate	ft_resource	Cannot write persistent ListDirectory data.
ft_listElementUnknown	ft_error	The schema and Ftscript do not match. There is no such element in a list!
ft_listenerCantClose	ft_resource	Listener could not be terminated.
ft_lockNotReadable	ft_resource	Interpreter process not accessible.
ft_logfileCantCreate	ft_resource	Log file could not be created. No logfile, no replay.
ft_logfileCantRead	ft_resource	Log file cannot be read. Replay not possible.
ft_logfileError	ft_resource	Log file could not be created. No logfile, no replay.
ft_logfileInvalid	ft_resource	Invalid log file.

Internal code	Error code	Description
ft_logfileIoError	ft_resource	IO error for log file.
ft_logfileNoSuchFile	ft_resource	Log file could not be found.
ft_logFileNotFound	ft_resource	Log file could not be found.
ft_logfileNotReadable	ft_resource	Log file cannot be read. Replay not possible.
ft_logfileWrongVersion	ft_error	Incorrect version of the log file. This request cannot be restarted.
ft_loggerCantClose	ft_resource	Logger could not be terminated.
ft_logWrapperCantCreate	ft_resource	Log file wrapper could not be created.
ft_logWrapperCantFind	ft_resource	Log file wrapper could not be found.
ft_logWrapperCantWrite	ft_resource	Log file wrapper could not be written.
ft_mainLockError	ft_resource	Main lock cannot be set. Interpreter will be terminated.
ft_mainLockIOerror	ft_resource	Main lock cannot be set. Interpreter will be terminated.
ft_mainLockNotSet	ft_resource	Main lock not set. Interpreter will be terminated.
ft_nameGroupElementUnknown	ft_error	The schema and Ftscript do not match. There is no such name attribute.
ft_nasty_error	ft_error	An unhandled error has occurred.
ft_noInterpreter	ft_resource	Server process not present at the correct time.
ft_noKeyInLogfile	ft_error	Log file entry invalid. "key" not present. Restart no longer possible.
ft_noPartner	ft_paramError	No partner was specified for the remote file.
ft_noRef	ft_reference	Reference cannot be resolved.
ft_noRefForContextObject	ft_reference	Reference cannot be resolved.
ft_noRefInFtscript	ft_reference	Reference cannot be resolved.

Internal code	Error code	Description
ft_noScript	ft_resource	No script was specified.
ft_noSession	ft_error	*open*FT session could not be generated.
ft_notExist	ft_notExist	File or directory does not exist.
ft_noTransferAdmission	ft_paramError	The transfer admission is missing in the partner specification.
ft_notRepeatable	ft_error	Ftscript request cannot be repeated.
ft_noUserFaultCode	ft_paramError	Error code may not start with 'ft'.
ft_noUserScript	ft_paramError	No script was specified for executeScript.
ft_noUtf8Support	ft_resource	UTF 8 is not supported.
ft_openParamError	ft_error	Error generating the *open*FT session.
ft_openParamError2	ft_error	ft_open was not successful, parameter error.
ft_OrderQueueNotFound	ft_resource	Order queue could not be found.
ft_parseByteArray	ft_error	Parse or IO error.
ft_parseError	ft_error	Parse or IO error.
ft_portFileCantClose	ft_resource	PortFile could not be closed correctly.
ft_portNotFound	ft_resource	No port found. Execution later.
ft_readStatus	ft_error	Parameter error while determining status.
ft_recordSize	ft_error	Incorrect specification for maxRecSize. Defective schema?
ft_recoveryCreateDirectory	ft_recoveryFailed	Recovery failed. See manual .
ft_recoveryFailed	ft_recoveryFailed	Recovery failed. See manual .
ft_replayIllegalEntry	ft_error	Neither an error code or request ID found on replay.
ft_reqlistOutOfRange	ft_error	Error querying the *open*FT request list.

Internal code	Error code	Description
ft_reqstatParamError	ft_error	Incorrect parameter for reqstat. Internal error.
ft_requestIdFormat	ft_error	Error in the format of the request ID.
ft_requestIdIllegalFormat	ft_error	Error in the format for the request ID.
ft_requestInvalid	ft_error	Invalid request.
ft_resumeForbidden	ft_error	Resume not possible.
ft_resumeUnloggedTransfer	ft_resource	More than one file transfer open.
ft_rformIllegal	ft_error	Incorrect specification for RecordFormat. Defective schema?
Ft_rformOutOfRange	ft_error	Incorrect specification for RecordFormat. ftAPI version incorrect?
ft_schemaConflict	ft_error	Unknown activity, schema defective?
ft_schemaNotFound	ft_resource	Schema file *ftscript.xsd* not installed.
ft_scriptElementUnknown	ft_error	The schema and Ftscript do not match. There is no such script.
ft_scriptIdNotGiven	ft_error	Script ID not specified.
ft_scriptinfoFileCantCreate	ft_resource	Script information cannot be generated.
ft_scriptinfoFileCantWrite	ft_resource	Script information cannot be written.
ft_scriptinfoFileCantWrite2	ft_resource	Script information cannot be generated.
ft_serverSignalCommand	ft_error	Error in request acceptance by server process.
ft_sessionNotFound	ft_resource	No valid *open*FT session found.
ft_signalingFailed	ft_resource	Signalling to port failed. Execution later.
ft_socketClose	ft_resource	Socket cannot be closed.
ft_stateUnknown	ft_resource	Unknown status.

Internal code	Error code	Description
ft_statusCantRead	ft_resource	Status cannot be read. Request cannot be executed.
ft_statusCantWrite	ft_resource	Cannot write status.
ft_termSocketClose	ft_resource	Socket cannot be closed.
ft_timestampCantCreate	ft_resource	Timestamp could not be created.
ft_tmpFileDelete	ft_resource	Timestamp could not be deleted.
ft_tmpFileNotFound	ft_resource	Expected TmpFile was not found.
ft_traceCantCreate	ft_resource	Trace file could not be generated. No trace (deactivate trace!)
ft_traceFileCreateError	ft_resource	Trace file could not be generated. (Deactivate trace)
ft_traceFileNotFound	ft_resource	Trace file could not be found.
ft_traceSerialization	ft_error	Trace file corrupt.
ft_traceSyncFailed	ft_resource	Trace data could not be written to disk. Trace may not be complete.
ft_traceWrapperCantCreate	ft_resource	Trace file wrapper could not be generated.
ft_traceWrapperCreateError	ft_resource	Trace file wrapper could not be generated.
ft_traceWrapperNotFound	ft_resource	Trace file wrapper could not be found.
ft_transferFileIllegalSyntax	ft_error	remote/remote or local/local not permitted. Schema defective?
ft_transferMandatoryParam	ft_error	Parameter error on ft_transfer.
ft_transparentModeIllegal	ft_error	Incorrect specification in transparent mode. Defective schema?
ft_transparentModeOutOfRange	ft_error	Incorrect specification in transparent mode. ft-API version correct?
ft_unexpectedLogEntry	ft_error	Log file invalid. Expected entry not found.

Internal code	Error code	Description
ft_unknownError	ft_error	Unknown error code.
ft_unlockOrderQueue	ft_resource	Order queue could not be released.
ft_unresolvedTmpFile	ft_reference	Local TmpFile was not found. Reference cannot be resolved.
ft_unwrap	ft_error	Unwrapping failed.
ft_userFault	ft_userFault	Error activity called in the Ftscript.
ft_userStorageCantCreate	ft_resource	Basic data cannot be initialized. EXIT, NO ERROR LOG!
ft_validatingFeaturesNot Supported	ft_error	The parser does not support necessary features for validation. Should NEVER occur!
ft_validationError	ft_error	Validation, parse or IO error.
ft_workDirectoryPiNotFound	ft_error	PI for call directory not found in script.
ft_workDirUnusable	ft_resource	Working directory cannot be generated.
ft_writeModeIllegal	ft_error	Incorrect specification in write mode. Defective schema?
ft_writeModeOutOfRange	ft_error	Incorrect specification in write mode. Defective schema?
ft_writeOrderQueue	ft_resource	Write to order queue failed.
ft_wrongJavaVersion	ft_resource	The current Java version must be lower than 1.4.2_07.

The following error codes are permitted for the *faulthandler* (see section “faulthandler” on page 62).

Error codes	Description
ft_abort	Command aborted.
ft_access	Access error.
ft_admin	Administration error
ft_auth	Authentication error.
ft_cantCreate	File/directory could not be generated.
ft_cantDelete	File/directory could not be deleted.
ft_configuration	Configuration defective.
ft_connection	Connection error.
ft_corrupt	File/data/directory cannot be used.
ft_error	General error.
ft_exist	File/directory already exists.
ft_execute	Error executing executeScript.
ft_local	The command script was terminated in the local system with an error.
ft_localFileStructure	Error in the local file.
ft_notEmpty	Directory is not empty.
ft_notExist	File/directory does not exist.
ft_notSupported	The functionality is not supported.
ft_panic	Severe internal error.
ft_paramError	Parameter error.
ft_paramTooLong	Parameter too long.
ft_recoveryFailed	Restart failed.
ft_reference	An Ftscript object which is not present is referenced.
ft_remote	The command script was terminated in the remote system with an error.
ft_remoteFileStructure	Remote file structure defective.
ft_resource	Resource error (e.g. not enough storage space, no RAM)
ft_script	A script executed with executeScript has terminated with an error code.
ft_syntax	Syntax error.

Glossary

Cross-references are written in *italics*.

Activity

An *openFT script* consists of activities. These can be operating instructions to *open*FT (e.g. *transferFile*, *deleteFile*) or statements which control the processing flow (e.g. *parallel*, *foreach*).

Context object

Context objects are described in an *activity*'s context. Each context object possesses an ID which must be unique in the context. The context object is addressed (referenced) via this ID.

file transfer request

FT request

FT request

see *open*FT request

FTAC (File Transfer Access Controll)

Part of *open*FT that offers extended access protection for file transfer and file management.

FTAM-1

document type for text files

FTAM-3

document type for binary files

FTAM file attributes

All systems which permit file transfer via FTAM protocols must make their files available to their partners using a standardized description (ISO 8571). To this end, the attributes of a file are mapped from the physical filestore to a *virtual filestore* and vice versa. This process distinguishes between three groups of file attributes:

– kernel group: describes the most important file attributes.
– storage group: contains the file's storage attributes.
– security group: defines security attributes for file and system access control.

FTAM partner

Partner system which uses the *FTAM protocols* for communication.

.ftsc

File name suffix for an *openFT-Script* file.

Interpreter

Here, the program which executes an *openFT-Script request*. The interpreter controls automatic *restarts*.

job

Sequence of commands, statements and data.

local system

The *FT system* at which the user is working.

***open*FT partners**

Partner systems which communicate via *open*FT *protocols*.

***open*FT request**

Request to an *FT system* to transfer a file from a *send system* to a *receive system* and possible start *follow-up processing requests*.

***open*FT script**

Name of the product and the langage in which *openFT requests* are formulated.

_open_FT script request

The processing of an *openFT script*.

owner of an FT request

Login name in the *local system* or *remote system* under which this *FT request* is executed. The owner is always the ID under which the request is submitted, not the ID under which it is executed.

partner system

here: *FT system* that executes FT request together with the *local system*.

password

Sequence of characters that a user must enter in order to access a user ID, file, job variable, network node or application. The user ID password serves for user *authentication*. It is used for access control. The file password is used to check access rights when users access a file (or job variable). It is used for file protection purposes.

remote system

see *Partner system*

request

here: *openFT request*

request queue

File which contains the *asynchronous requests* and their processing states. The request queue also contains the parameters set with the *ftmodo* command.

restart

Automatic continuation of an *FT request* after an interruption.

system, remote

see *partner system*

system, local

see *local system*

TCP/IP (Transmission Control Protocol/Internet Protocol)

Widespread protocol for file transfer (corresponds roughly to Layers 3 and 4 of the OSI Reference Model, i.e. Network and Transport Layer); was originally developed for the ARPANET (computer network of the US Ministry of Defense, now a de-facto standard.

Trace

Diagnostic function which logs the course of FT operation.

Transmission Control Protocol/Internet Protocol

see *TCP/IP*

Transport Name Service (TNS)

Service used to administer properties specific to transport systems. Entries for *partner systems* receive the information on the particular *transport system* employed.

Index

A
activities 29, 30
  external 30
  internal 30
  parent and child 30
attributes 34
  directory names 35
  file name 34
AutoDataSpec 94

B
baseDir 44

C
case 64
comment 45
components of an *open*FT script 29
context 29, 31, 46
createDirectory 47

D
default 64
deleteDirectory 49
deleteFile 51
de-referencing 31
  example 32
  rules 31
diagnostic information 41
directory 53
directory name attributes 35

E
empty 54
end of an *open*FT script 40
error codes 36, 111
  normal 36, 37
  permitted for faulthandler 137
  severe 36, 38
error handling 29, 36
error in an *open*FT script 40
error messages 36, 111
executeScript 56

F
fault 59
faulthandler 29, 31, 62
  case 64
  default 64
  permitted error codes 137
file 66
file name attributes 34
foreach 67
fromLocalFile 97
fromLocalTmpFile 99
fromRemoteFile 102
ftacAdmission 82
ftscript 29, 73
ftscript error codes 36

I
initialization 40

L
list 74
listDirectory 75

N
normal error codes 36, 37
notational conventions 9

O
*open*FT script
  components 29
  diagnostic information 41
  end 40
  error 40
  execution 40
  initialization 40
  running 40
  structure 29

*open*FT-Script statements 43
attributes 34
baseDir 44
comment 45
context 46
createDirectory 47
deleteDirectory 49
deleteFile 51
directory 53
directory name attributes 35
empty 54
error handling 36
executeScript 56
fault 59
faulthandler 62
file 66
file name attributes 34
foreach 67
ftscript 73
list 74
listDirectory 75
overview 43
parallel 78
partner 80
script 84
sequence 86
transferFile 88

P
parallel 78
partner 31, 80
ftacAdmission 82
processingAdmission 81
transferAdmission 82
userAdmission 82
processingAdmission 81

R
referencing 29, 31
remoteFailureScript 104
remoteSuccessScript 106
request queue 141
restart 39
root element 29, 30
root element ftscript 29
running an *open*FT script 40

S
script 84
script statements
see *open*FT-Script statements
sequence 86
severe error codes 36, 38
specifying directory names 35
specifying the file name 34
statements, see *open*FT-Script statements
structure of an *open*FT script 29

T
target group 7
TCP/IP 142
toLocalFile 107
toLocalTmpFile 108
toRemoteFile 110
transferAdmission 82
transferFile 88
AutoDataSpec 94
fromLocalFile 97
fromLocalTmpFile 99
fromRemoteFile 102
remoteFailureScript 104
remoteSuccessScript 106
toLocalFile 107
toLocalTmpFile 108
toRemoteFile 110

U
userAdmission 82

Fujitsu Siemens Computers GmbH
User Documentation
81730 München
Germany

Fax: 0 700 / 372 00000

email: manuals@fujitsu-siemens.com
http://manuals.fujitsu-siemens.com

Comments
Suggestions
Corrections

Submitted by

Comments on *open*FT V10.0 for UNIX Systems and Windows Systems
*open*FT-Script Interface

Information on this document

On April 1, 2009, Fujitsu became the sole owner of Fujitsu Siemens Computers. This new subsidiary of Fujitsu has been renamed Fujitsu Technology Solutions.

This document from the document archive refers to a product version which was released a considerable time ago or which is no longer marketed.

Please note that all company references and copyrights in this document have been legally transferred to Fujitsu Technology Solutions.

Contact and support addresses will now be offered by Fujitsu Technology Solutions and have the format *...@ts.fujitsu.com*.

The Internet pages of Fujitsu Technology Solutions are available at *http://ts.fujitsu.com/*...
and the user documentation at *http://manuals.ts.fujitsu.com*.

Copyright Fujitsu Technology Solutions, 2009

Hinweise zum vorliegenden Dokument

Zum 1. April 2009 ist Fujitsu Siemens Computers in den alleinigen Besitz von Fujitsu übergegangen. Diese neue Tochtergesellschaft von Fujitsu trägt seitdem den Namen Fujitsu Technology Solutions.

Das vorliegende Dokument aus dem Dokumentenarchiv bezieht sich auf eine bereits vor längerer Zeit freigegebene oder nicht mehr im Vertrieb befindliche Produktversion.

Bitte beachten Sie, dass alle Firmenbezüge und Copyrights im vorliegenden Dokument rechtlich auf Fujitsu Technology Solutions übergegangen sind.

Kontakt- und Supportadressen werden nun von Fujitsu Technology Solutions angeboten und haben die Form *...@ts.fujitsu.com*.

Die Internetseiten von Fujitsu Technology Solutions finden Sie unter *http://de.ts.fujitsu.com/*..., und unter *http://manuals.ts.fujitsu.com* finden Sie die Benutzerdokumentation.

Copyright Fujitsu Technology Solutions, 2009